*electronics*

MDPI

*Article*

# A Generic Interface Enabling Combinations of State-of-the-Art Path Planning and Tracking Algorithms

**Johannes Rumetshofer [1,2,*], Michael Stolz [1,2] and Daniel Watzenig [1,2]**

[1] Institute of Automation and Control, Graz University of Technology, 8010 Graz, Austria; michael.stolz@tugraz.at (M.S.); daniel.watzenig@tugraz.at (D.W.)
[2] Virtual Vehicle Research GmbH, 8010 Graz, Austria
* Correspondence: j.rumetshofer@tugraz.at

**Abstract:** In the development of Level 4 automated driving functions, very specific, but diverse, requirements with respect to the operational design domain have to be considered. In order to accelerate this development, it is advantageous to combine dedicated state-of-the-art software components, as building blocks in modular automated driving function architectures, instead of developing special solutions from scratch. However, e.g., in local motion planning and control, the combination of components is still limited in practice, due to necessary interface alignments, which might yield sub-optimal solutions and additional development overhead. The application of generic interfaces, which manage the data transfer between the software components, has the potential to avoid these drawbacks and hence, to further boost this development approach. This publication contributes such a generic interface concept between the local path planning and path tracking systems. The crucial point is a generalization of the lateral tracking error computation, based on an introduced error classification. It substantiates the integration of an internal reference path representation into the interface, to resolve the component interdependencies. The resulting, proposed interface enables arbitrary combinations of components from a comprehensive set of state-of-the-art path planning and tracking algorithms. Two interface implementations are finally applied in an exemplary automated driving function assembly task.

**Keywords:** ODD-based AD function design; path tracking; path planning; software architecture; interface design



**Citation:** Rumetshofer, J.; Stolz, M.; Watzenig, D. A Generic Interface Enabling Combinations of State-of-the-Art Path Planning and Tracking Algorithms. *Electronics* **2021**, *10*, 788. https://doi.org/10.3390/electronics10070788

Academic Editor: Soon Ki Jung

Received: 24 February 2021
Accepted: 23 March 2021
Published: 26 March 2021



## 1. Introduction

Automated driving (AD) has been a huge challenge over the last decades in research as well as in industry. However, estimations and expectations for final breakthroughs on the market have been continuously shifted, contrary to many announcements. An analytic look at the available products on the market reveals that SAE Level 2 [1] is still state-of-the-art in passenger cars, outdone in few applications to Level 3 for very restrictive applications (e.g., [2]). Level 4 automation, in particular for urban road networks, is still an open challenge (see [2]) and therefore in focus of current research. From a general perspective, the step from Level 3 to 4 is a shift from automated to autonomous driving. This step involves full environmental perception and decision making. Furthermore, the human driver can not be applied as safety fallback anymore. Consequently, it is a game changing development step. From implementation point of view a paradigm shift in system architecture is required since sensors must be shared between components and new concepts for multi-layer control software are required.

In contrast to the "under all conditions" requirement of Level 5 automation, which is quite problematic from a technical perspective, Level 4 automation solves the task of fully autonomous driving under clearly defined conditions, i.e., in a specified operational design domain (ODD). Different ODDs can be very diverse, e.g., valet parking and highway driving, and, hence, require a carefully matched AD function. From this point of view, a

modular system architecture design (e.g., [3]) is beneficial ([2]). It enables the development of modular and, hence, reusable software components with respect to dedicated tasks of the fundamental sense-plan-act principal from robotics [4], and ODD-based assembly and tuning of these components. An attempt to overview the scientific state-of-the-art on all stages of the sense-plan-act principal, reveals a tremendous amount of highly sophisticated scientific solutions (see, for example, [5,6], (pp. 71–140) and [7]), with promising simulation and also real-world testing results in specific use cases. Although equivalent components solve the same task on a qualitative level, they diverge in their specific input/output data requirements. This fact complicates ODD-based AD function assembly, yielding potential performance interdependencies between the components. This has to be avoided since the components form a safety critical overall system. Hence, a clear strategy on the definition and implementation of component interfaces is beneficial. A common approach is to define very basic, static interfaces (cf. Autoware [8]), focusing on a minimized version of the shared information. The drawback of this approach is the need for repetitive wrapper code within the single software components adapting the interface signals to match the components' requirements. Although this wrapper code is independent from the actual component algorithm, it eventually impacts the performance of the component and of the overall combined system. A modification that affects this code involves all components. In order to overcome this drawback, these wrapper-tasks may be assigned to a dedicated interface component following, if possible, a generic design approach. This enables a clear focus of the connected components on their dedicated tasks omitting component independent input/output data processing. Simultaneously, the application of dedicated interface components supports a middelware independent component development and reduces later integration risk in combination with different middleware concepts.

The design of such an interface has to handle manifold requirements of state-of-the-art components. This obviously, on the one hand, complicates the interface design to a challenging task. On the other hand, it reveals significant benefits especially in the ODD-based AD-function assembly, since it enables straight-forward combination of arbitrary state-of-the-art components.

This publication is dedicated to the design of such an interface, between the local motion planning system and the path tracking system, which is an essential part of every AD function. The interface, hence, shall be able to connect a comprehensive set of state-of-the-art path planning and path tracking algorithms (see Figure 1), resolving performance interdependencies to the greatest possible extend. This is of special interest for example in collision avoidance, see for example [9,10], since it requires an accurate coordination of path planning and tracking.

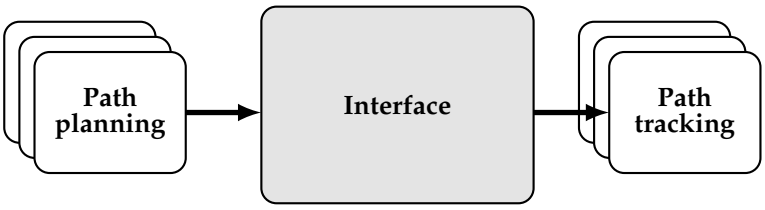


**Figure 1.** A generic interface shall enable arbitrary combinations of state-of-the-art path planning and tracking components, without any need for repetitive wrapper code for input/output data processing within the components.

The interface design needs an overview on state-of-the-art components and algorithms. Therefore, the publication simultaneously features a survey contribution, in particular with respect to path tracking algorithms (from a specific point of view) and specific interpolation methods. The presented classification of lateral tracking error definitions based on a comprehensive set of state-of-the-art tracking controllers is essential for the proposed interface design, but also contributes to a better understanding of the impact of tracking error definition on the performance of a tracking controller. The proposed interface design,

furthermore, may serve as conceptual model for the design of other interfaces in modular system architecture for AD functions.

In detail, the publication structures as follows: Section 2 outlines the general path tracking problem and identifies the lateral tracking error computation as a major challenge in the generic interface design. In order to tackle this challenge, Section 3 introduces a lateral tracking error classification approach. This proposed classification enables a generalized approach in tracking error computation (see Section 4), based on a concise set of three elementary path operations. This set is sufficient to cover a comprehensive set ot state-of-the-art path tracking controllers. Section 5 summarizes the contribution of the preliminary findings for the aimed interface design. Section 6 extends the interface requirements from the path planning side, yielding the final interface design concept in Section 7. Finally, in Section 8 a simple, exemplary ODD-based AD function assembly tasks is considered in order to demonstrate the proposed interface design and to substantiate its benefits. Appendixs A and B provide continuative theoretical basics on path parametrization and interpolation.

## 2. The Path Tracking Problem

Path tracking is an important sub-problem of motion control in automated driving tasks. In contrast to the more general trajectory tracking (see for example [11,12], (p. 172)) the trajectory is split into a time-independent, spatial information—the reference path—and a time-dependent function, which defines the reference position at a specific time along this path. A lateral controller applies steering commands to approach the reference path—the so-called path tracking. Concurrently a longitudinal controller applies vehicle speed adaptions in order to track the time-dependent reference position along the path. The basic assumption behind a separated lateral and longitudinal control is almost decoupled lateral and longitudinal vehicle dynamics. In fact, they are actually coupled due to the limited traction forces of the vehicle tires and the time-dependent actuation limitations (steering rate). Whereas this assumption is valid in low- and moderate-speed driving at dry roads, respectively in high-speed driving with low steering dynamics, it is not valid in highly dynamic maneuvers, like emergency evasions or when driving on an icy road.

There exists several surveys on the classification of state-of-the-art tracking controllers like [5,13–15] and [6] (pp. 71–140). In general, the path tracking problem can be stated as follows: Given an planar reference path, which might be represented in a parametric way (cf. Appendix A) with respect to a curve parameter $\tau$,

$$\boldsymbol{\gamma}(\tau) = \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^T, \tag{1}$$

the path tracking controller has to compute a steering actuation, e.g., a steering wheel angle, in order to make a the position of the vehicle to follow the reference path. In order to use classical control system design approaches an appropriated error vector has to be defined,

$$\mathbf{e}(t) = \mathbf{f}_\mathrm{e}(\boldsymbol{\gamma}(\tau), \mathbf{p}(t)) = \begin{bmatrix} e_\mathrm{lat}(t) & e_\psi(t) & \cdots \end{bmatrix}^T, \tag{2}$$

based on the reference path and the vehicle pose. The vehicle pose consists of the vehicle position $x(t), y(t)$ and heading $\psi(t)$ and possibly more vehicle states,

$$\mathbf{p}(t) = \begin{bmatrix} x(t), y(t), \psi(t), \ldots \end{bmatrix}^T. \tag{3}$$

The error vector consists of a mandatory spatial tracking error (some lateral offset error) and optionally of additional error measures like an orientation error or curvature error. Some special control approaches, e.g., Model Predictive Control (MPC), furthermore, require several lateral tracking errors corresponding to vehicle pose predictions.

With respect to a specific control error, the path tracking controller defines a steering command $\delta(t)$, which shall control the tracking error to zero, and consequently make the vehicle to follow the reference path:

$$\delta(t) = f_\delta(\mathbf{e}(t)): \quad \mathbf{e}(t) \to \mathbf{0} \tag{4}$$

There are various reasonable possibilities to the define a lateral tracking error (see Figure 2) based on different motivations. In general, there exists no unique mappings between different lateral errors. It is an important but little-noticed fact that both the error definition and the control law design have to be considered as degrees of freedom in control system design and impact the final tracking performance.

The computation of the tracking error depends on the reference path, which is provided by the motion planning system. Consequently, its constitution and quality are defined by another component, which yields an undesired dependency on the planning system. Therefore, it is reasonable to think about outsourcing the tracking error computation to a generic interface component. To do so, it is necessary to analyze state-of-the-art path tracking algorithms with respect to the applied lateral tracking error definition. A unique classification of the used tracking error definitions and a generalization of the tracking error computation based on this classification is a major step towards the intend generic interface, which is addressed in the next sections.

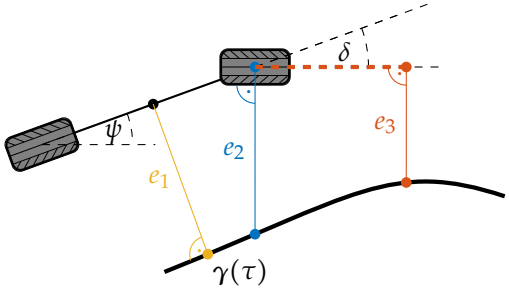


**Figure 2.** Different tracking error definitions of state-of-the-art tracking controllers.

## 3. Tracking Error Classification

Based on the analysis of a comprehensive set of state-of-the-art path tracking controllers (cf. Table 2), the introduction of three general and comprehensible classifier sets are introduced ($S_{\text{vr}}$: vehicle reference, $S_{\text{lh}}$: look-ahead (direction and distance) and $S_{\text{eo}}$: error orientation), which are applicable to classify these controllers and to analyze their differences.

$$S_{\text{SotA}} \in S_{\text{vr}} \times S_{\text{lh}} \times S_{\text{eo}} \tag{5}$$

These classifiers shall now be discussed in more detail, in order to summarize the corresponding motivation and the effects on the control design and control performance.

### *3.1. Vehicle Reference Point*

The vehicle reference might be defined at an arbitrary point on the vehicle chassis. There is a set of common reference points, based on a single-track abstraction of a front-axle-steered vehicle (see Figure 3) with different motivations:

- Rear axle: A vehicle, in general, features non-holonomic dynamics. The rear axle is the point of the vehicle with the "most constrained" motion. Assuming zero lateral slip, the motion of the rear axle is aligned with the vehicle heading. Therefore, a constant zero tracking implies that also the vehicle heading is aligned to the reference path, which is a favorable tracking property. From control system theory the center of the rear axle is of interest, as it is a flat output of the system restricting on slip-free vehicle kinematics (see for example [11,16]). The turning radius of the rear axle in cornering is smaller than the turning radius of the front axle (see Figure 4). Therefore, the choice of the rear axle as a vehicle reference point, in general, implies potential undesired overshooting of the vehicle's front.
- Front axle: If the vehicle reference point is set to the center of the front axle, the non-holonomic vehicle kinematics in principle do not have to be considered in the

control design, as stopping and adjustment of the steering angle enables tracking of arbitrary reference paths within the limited turning radius. This enables a simplified control design, especially for low dynamic driving tasks as parking. A drawback of this reference point is the smaller turning radius of the rear axle in cornering (cf. rear axle reference point), which implies potentially undesired curve cutting.

- Center of gravity: The choice of the center of gravity as a vehicle reference, simplifies the setup of the vehicle's equations of motion. Therefore, it is used in many control system design approaches. From tracking perspective its position, somewhere in the middle of the car is of interest, in order to minimize the total distance of all points with respect to the reference path.
- Center of oscillations/percussion: In the center of oscillation or percussion, the translation and rotation impact of a lateral tire slip at the rear axle are in balance. Consequently, this point is of special interest in order to design control laws, which are robust with respect to lateral rear axle tires slip. The choice of this reference point is popular in tracking controllers designed for limit-handling, as racing applications (see for example [17,18]). For front-wheel-steered vehicles the position of the center of percussion with respect to the rear axle $l_{\mathrm{CP}}$ is:

$$l_{\mathrm{CP}} = l_{\mathrm{CG}} + \frac{I_{\mathrm{zz}}}{m \times l_{\mathrm{CG}}}, \tag{6}$$

where $l_{\mathrm{CG}}$ is the distance of rear axle and center of gravity, $m$ the vehicle mass and $I_{\mathrm{zz}}$ the vehicle's inertia in the center of gravity with respect to the vertical vehicle axis.

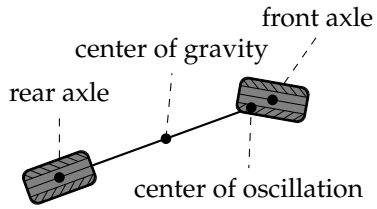


**Figure 3.** Common vehicle reference points for control error definition based on a single-track vehicle model.

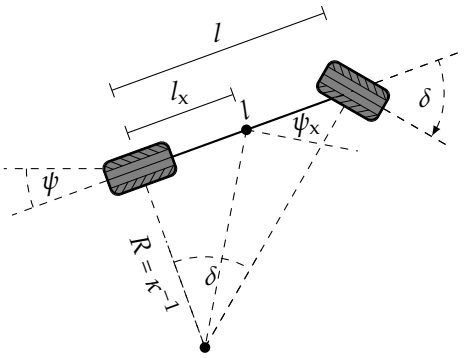


**Figure 4.** Direction of motion $\psi_x$ of different reference points.

The set of vehicle reference classifiers $S_{\mathrm{vr}}$, hence, can be summarized as:

$$S_{\mathrm{vr}} = \{\mathrm{rear}, \quad \mathrm{front}, \quad \mathrm{CG}, \quad \mathrm{CP}\}. \tag{7}$$

### *3.2. Look-Ahead*

Due to the non-holonomic motion of a vehicle, reference points on the path parallel to the vehicle are out of reach without reverse driving or spacious maneuvers. Therefore, many tracking error definitions do not directly use the vehicle reference point to compute a lateral tracking error, but some point ahead of the vehicle reference point. The application

of such a so-called look-ahead, or preview (see, for example, [19]), is a very natural behavior of human drivers. The look-ahead refers to the vehicle reference point and is defined by a look-ahead direction and a look-ahead distance. The introduction of a look-ahead enables preventive reaction to sudden direction changes of the reference path, which stabilizes the vehicle motion. As this becomes more crucial for higher vehicle speeds several tracking controllers (see, for example, [19–22]) use an adaptive, velocity dependent look-ahead distance, instead of a fixed distance. This is advantageous also if the tracking controller is applied in combination with simple motion planning systems, which do not provide a smooth path (cf. Section 8), since the look-ahead damps the impact of path discontinuities. In [23], the impact of this damping characteristic is analyzed with respect to control stability in frequency domain base on a linearized vehicle model. This damping characteristic at the same time reveals the main drawback of a look-ahead application. A smooth vehicle motion is achieved at the cost of worse tracking performance in the vicinity of the actual vehicle position (e.g., curve cutting). Even in the case of perfect reference tracking $e_{\mathrm{lat}}(t) \equiv 0$ the vehicle itself actually does not follow the planned reference path. This implies that planned reference path properties, like specific safety distances to obstacles, are withdrawn. If a tracking controller is applied in combination with a comprehensive motion planning system, consequently, one should carefully think about the application of a look-ahead in the tracking error definition. Three reasonable choices for the direction of a look-ahead appear (see Figure 5):

- look-ahead towards the vehicle heading,
- look-ahead towards the direction of motion in vehicle reference point,
- and look-ahead towards the reference path in a certain distance.

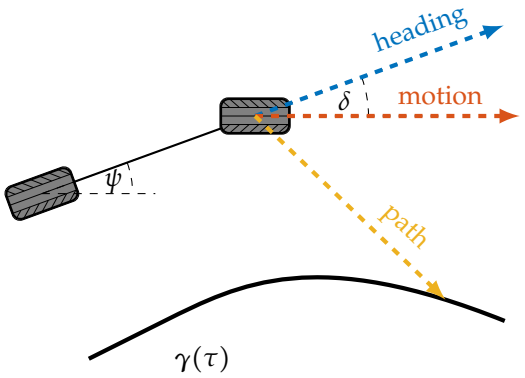


**Figure 5.** Different look-ahead directions based on the vehicle reference point in the center of the front axle.

Considering vehicle kinematics (see Figure 4), the absolute direction of motion $\psi_{\mathrm{x}}$ in a vehicle reference point at distance $l_{\mathrm{x}}$ in front of the rear axle (towards the vehicle heading $\psi$), with respect to a steering angle $\delta$ is:

$$\psi_{\mathrm{x}} = \psi + \arctan\left(\frac{l_{\mathrm{x}}}{l}\tan\delta\right). \tag{8}$$

This direction is obviously equal to the vehicle heading at the rear axle (for $l_{\mathrm{x}} = 0$) and equal to the steering direction at the front axle ($l_{\mathrm{x}} = l$). The application of a look-ahead extends the possibilities in achieving specific constitution of the final model-based tracking problem, as it is proposed for example in [12] (pp. 199–201) (front axle vehicle reference and a look-ahead in direction of motion): In this case, decoupling and input–output linearization can be achieved with a static feedback. The above considerations yield the set of look-ahead classifiers $S_{\mathrm{lh}}$:

$$S_{\mathrm{lh}} = \{\text{heading}, \quad \text{motion}, \quad \text{path}\}. \tag{9}$$

### *3.3. Error Orientation*

Based on a vehicle reference point and the optional look-ahead (direction and distance) the orientation of the tracking error needs to be defined in order to compute a lateral tracking error. Similar to the definition of look-ahead directions, reasonable orientations are:

- perpendicular to the vehicle heading,
- perpendicular to the direction of vehicle motion in the vehicle reference point,
- perpendicular to the path.

For almost straight reference paths, different error orientations approximately coincide if the vehicle drives along the reference path. On the other hand, if the vehicle first has to approach the reference path, or drives along curvy path sections, differences reveal (see Figure 6). In summary the classification set for the error orientation is:

$$S_{\text{eo}} = \{\text{heading}, \quad \text{motion}, \quad \text{path}\}. \tag{10}$$

The above definitions can be used to give a comprehensive definition of a lateral tracking error based on the classifiers *vehicle reference*, *look-ahead* (direction and distance) and *error orientation* in the combined set,

$$S = S_{\text{vr}} \times S_{\text{lh}} \times S_{\text{eo}}, \tag{11}$$

according to (7), (9) and (10). Figure 6 exemplarily illustrates all possible error definitions for the vehicle reference at the front axle, i.e., for the set $\{\text{heading}\} \times S_{\text{lh}} \times S_{\text{eo}}$, as listed in Table 1. In the case of a look-ahead towards the path and an error orientation perpendicular to the path, the lateral error has to be defined either with respect to the vehicle heading or motion (see errors $e_{\text{p,p(h)}}$ and $e_{\text{p,p(m)}}$).

Based on given coordinates of the vehicle reference point, a specified look-ahead distance and direction and error orientation define a reference point on the path. In addition to the lateral tracking error, this reference point can be used to compute further error measures like an orientation or curvature error. In [24], the choice of an appropriate heading error is discussed and its impact on the tracking performance is analyzed.

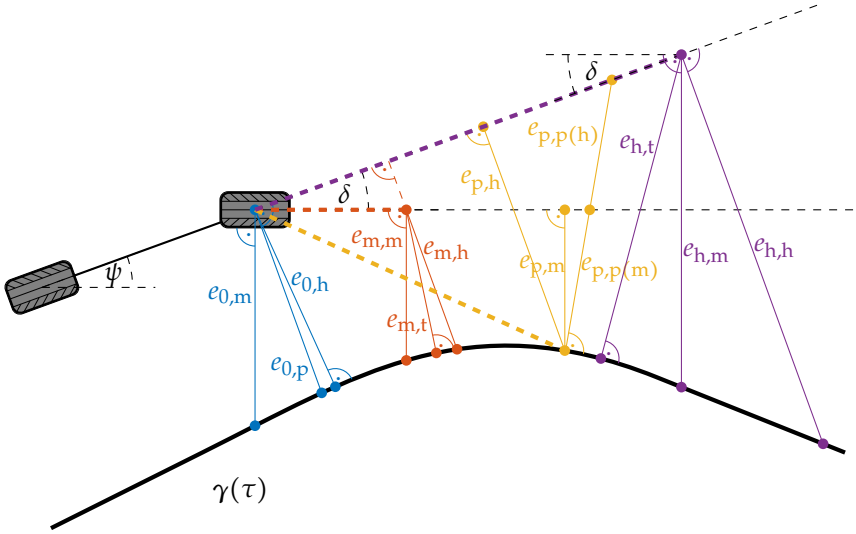


**Figure 6.** Possible tracking error definitions for vehicle reference at the front axle (see Table 1), i.e., the classification set $\{\text{front} \times S_{\text{lh}} \times S_{\text{eo}}\}$.

**Table 1.** Corresponding tracking error definition classifiers to Figure 6.

| | Look-Ahead | Error Orientation |
|---|---|---|
| $e_{0,h}$ | no | heading |
| $e_{0,m}$ | no | motion |
| $e_{0,p}$ | no | path |
| $e_{h,h}$ | heading | heading |
| $e_{h,m}$ | heading | motion |
| $e_{h,p}$ | heading | path |
| $e_{m,h}$ | motion | heading |
| $e_{m,m}$ | motion | motion |
| $e_{m,p}$ | motion | path |
| $e_{p,h}$ | path | heading |
| $e_{p,m}$ | path | motion |
| $e_{p,p(h)}$ | path | path (heading) |
| $e_{p,p(m)}$ | path | path (motion) |

### *3.4. Application to State-of-the-Art Tracking Controller*

To prove the applicability of the proposed tracking error definition classification, it is applied to a comprehensive set of state-the-art tracking controllers. Table 2 lists the classification according to the specified classifiers.

**Table 2.** Tracking error classification of state-of-the-art path tracking controllers.

| Controller | Vehicle Ref. | Look-Ahead | Error Orient. |
|---|---|---|---|
| Hoffmann (*Stanley*) [25], Kolb [26] | front | no | path |
| Sun [27,28], Kritayakirane [17], Chen [29], Hu [30] Bruschetta [31] | CG | no | path |
| Chatzikomis [14], Xu [19], Zhang [32] | CG | no | heading |
| Hiraoka [33] | CP | no | path |
| Tieber [34], Samson [35], Dominguez [36], Solea [37] | rear | no | path |
| Nestlinger [20], Ackermann [38], ARGO [21], Guldner [23], Yuan [22] | CG | heading | heading |
| Elkaim [39] | CG | heading | path |
| Solea [37] | rear | motion | path |
| Sentouh [40] | CG | motion | motion |
| Coulter (*Pure-pursuit*) [41] | rear | path | heading |

While Table 2 shows the application of various vehicle reference points, and look-ahead directions, most tracking controllers are based on an error orientation perpendicular to the reference path or to the vehicle heading. Only one of the considered controllers [40] is based on an error orientation perpendicular to the vehicle motion in the reference point. A possible explanation for this is the fact that direct measurement of the actual vehicle motion direction is challenging due to the side slip of the tires. As an alternative, an estimation based on the yaw rate has to be used. In fact, in [40], the authors do not propose a steering controller for application in an autonomous vehicle but propose a human driver model for simulation purpose. In [42], a hybrid concept consisting of an Pure-pursuit and Stanley controller is proposed in order to combine a tracking error definition with look-ahead and without look-ahead. In [17], the control law is designed with a vehicle reference point in the center of percussion. The used lateral control error $e_{CP}$, however, is a projection of the distance of the center of gravity to the path $e_{CG}$:

$$e_{CP} = e_{CG} + d_{CG,CP} \times \sin e_{\psi}, \tag{12}$$

according to the distance between center of gravity and center of percussion $d_{CG,CP}$ and the heading error $e_\psi$. Therefore, the vehicle reference point is assigned to the center of gravity. In [27,28,31], an MPC is applied for path tracking. In every MPC step, actuation signals are optimized with respect to a specific cost function on a prediction horizon. In general, the cost function includes a deviation from the reference path, i.e., a lateral tracking error. Instead of an evaluation of the tracking error at a single vehicle pose (cf. Figure 8), hence additionally an evaluation at several predicted vehicle poses is required.

The choice of the single introduced classifiers can be considered as part of the control parametrization. As Figure 6 indicates, the actual control error computation is defined by specific geometrical operations with respect to the current vehicle pose, the control parametrization and the provided reference path. A generalization of the tracking error computation is presented in the next section.

## 4. Tracking Error Computation

Based on a generalized reference point, which already considers an optional vehicle-oriented look-ahead (except for a look-ahead towards to reference path), it is noticeable that despite the various possibilities in the tracking error definition the actual error computation can be handled with only five geometric operations (see Tables 3 and 4 and Figure 7).

The operations **a** (intersection of two lines) and **c** (projection of a point on a line) do not depend on the reference path and have simple analytic solutions. Contrary, the operations **A** (intersection of the reference path with a line), **B** (intersection of the reference path and a circle) and **C** (point projection onto the reference path) depend on the path representation (cf. Appendix A). Hence, the reference path representation also determines the resulting computational complexity. Some general statements can be made on this path operations.

**Table 3.** Path operations.

| | **Operation** |
|---|---|
| **A** | intersection line/path |
| **B** | intersection circle/path |
| **C** | point projection on path |
| **a** | intersection line/line |
| **c** | point projection on line |

**Table 4.** Operations for error computation.

| | | **Error Orientation** | | |
|---|---|---|---|---|
| | | **Path** | **Heading** | **Motion** |
| **look-ahead** | no | **C** | **A** | **A** |
| | path | **B+a** | **B+c** | **B+c** |
| | heading | **C** | **A** | **A** |
| | motion | **C** | **A** | **A** |

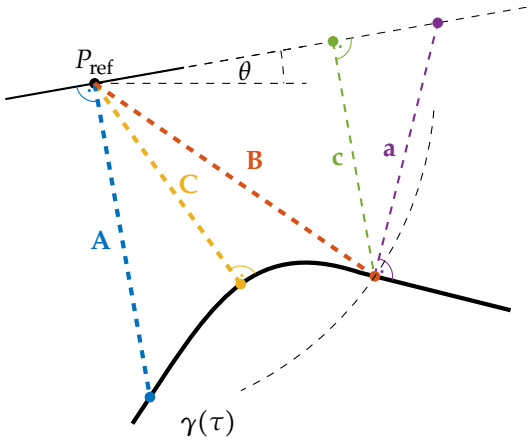


**Figure 7.** Visualization of path operations in the lateral tracking error computation.

### *4.1. Intersection of Reference Path and a Straight Line*

After performing a translation and rotation of the path, this operation reveals as common root finding problem, which can, in general, not be solved analytically, for example in case of a higher order polynomial reference path. However, many well-established numeric algorithms like Newton-method, bisection-method or Brent-method, can be used to compute arbitrarily exact approximations to the solution of this problem (see, for example, [43–45]). The solution of this path operation is neither unique nor it exists for sure. Hence, tracking controllers relying on this path operation inevitably have to define a fallback solution for the control law. The fallback solution, e.g., a constant curvature turn, has to ensure the convergence to a region where the path operation yields a solution. With respect to the uniqueness, it is reasonable to use the closest solution.

### *4.2. Intersection of Reference Path and a Circle*

From an analytic perspective, this operation can be considered equivalent to operation **A** after performing a transformation of the corresponding reference path section into polar coordinates, which is challenging for an arbitrarily parameterized path. From numerical perspective, it might also be considered as a generalization of the shortest-distance-problem (cf. operation **C**), searching for a fixed given distance to the reference path instead of the shortest distance. The solution of this path operation in general yields at least two solutions, which can be prioritized following again the most-progress-on-path principle. The solution of this path operation obviously does not exist if the shortest distance between the center point of the circle and the path exceeds the specified circle radius.While this offers a straight-forward method to test for the existence of a solution, there is, in contrast to operation **A**, no simple fallback, which can be applied for arbitrary controllers, which rely on this path operation. Therefore, a reliable fallback solution, which ensures that the vehicle is approaching the path, has to be design including the specific control law. For many controllers including the famous pure-pursuit controller [41], the solution of the shortest-distance-problem (see operation **C**) is applicable as a fallback solution.

### *4.3. Point Projection Onto the Reference Path*

Point projection is a well investigated topic, but still an ongoing research field in geometry (see, for example, [46–49]). Although for some path representations (e.g., if the path consists of circular arcs) an analytic solution exists, most implementations apply iterative numeric methods, similar to path operation **A**, to compute approximations of the solution. The existence of a solution is not ensured. A widely used fallback for tracking controllers is to transition to the shortest-distance-problem. A solution to this fallback problem always exists and if a solution to the point projection problem exists, the solutions are equivalent. The ambiguity of the solution can be handled based on the path direction (choosing the solution which gains the most progress along the reference path).

## 5. Summary of Interface Requirements from Tracking Control Perspective

The comprehensive investigations of the last sections provide the theoretical backbone in order to state the general requirements for a generic planning and control interface from control side. The central idea for the interface implementation is to detach the tracking error computation from the control component. On the one hand, this avoids redundant computations in different control components. On the other hand, it enables careful coordination with respect to the representation of the provided reference path. This supports the performance of the overall system, which is especially of interest in safety critical applications like collision avoidance. According to Section 3, the definition of three classifiers (vehicle reference point, look-ahead and error orientation) is sufficient to define the lateral tracking error definition applied in state-of-the-art path tracking controllers. Furthermore, according to Section 4, three elementary path operations are sufficient to compute all possible tracking errors, which can be defined based on the introduced tracking error classifiers. Hence, these elementary path operations can be implemented with respect to the reference path representation outside of the actual path tracking component, in order to supply various different tracking controllers with the required tracking error, as illustrated in Figure 8.

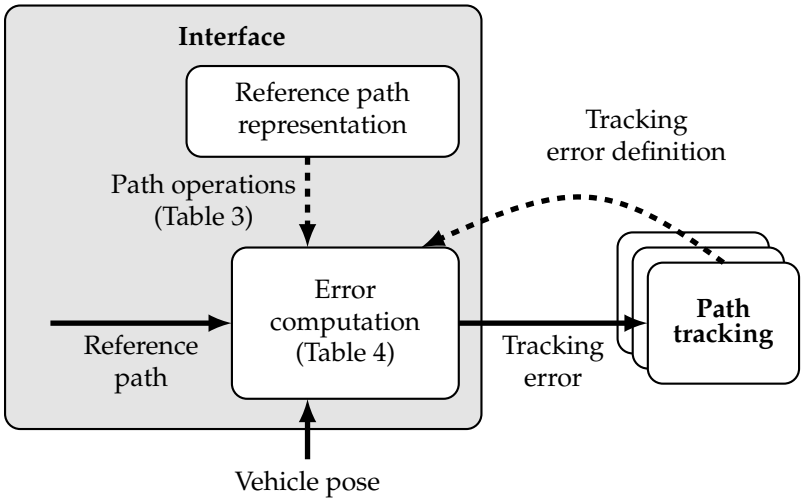


**Figure 8.** Interface concept to meet the output requirements defined by the state-of-the-art in path tracking controller design.

In order to complete the interface concept, the following section consider the path planning problem in general and the interface requirements resulting from the state-of-the-art in path planning.

## 6. Interface Requirements from Path Planning Perspective

Similar to path tracking path planning, is a sub-problem of the more general trajectory planning problem. It focuses exclusively on the spatial planning, also known as lateral planning, neglecting the temporal aspect of the motion planning. The general limitations of the applicability of such a decoupled motion planning and control have already been mentioned in the Section 2. In addition, from planning perspective also limitations of this approach with respect to complex dynamic environments arise. However, this separation enables simplified planning for relevant use-cases like valet parking. The general path planning problem can be stated as follows: The planner has to compute a collision-free path from a given starting pose to a given final pose, complying constraints with respect to drivability (limited actuation systems, like a vehicle's bounded steering angle), comfort (bounded lateral acceleration and jerk), efficiency (length of the path and necessary actuation effort) and safety (distance to obstacles). Many surveys on different path planning approaches and algorithms have been published (see, for example, [2,5,50–52]).

In contrast to the state-of-the-art path tracking controllers, where it was necessary to establish a reasonable generalization of the input requirements via a careful study of state-of-the-art components, the generalization of the output requirements of path planning components is more straight forward and does not need an extensive consideration of specific state-of-the-art path planning algorithms. On a qualitative level, there are two different output formats provided by path planning systems: Hermite path data and analytic curve expressions.

Hermite path data consist of a set of consecutive way points (position $x, y \mapsto$ G0 Hermite data) and optional higher-order geometric information: tangent (position $x, y$ + orientation $\psi \mapsto$ G1 Hermite data), curvature (position $x, y$s + orientation $\psi$ + curvature $\kappa$: $\mapsto$ G2 Hermite data (cf. Figure 9)) and so on. Hermite path data give a hint about the qualitative course of a continuous path, assuming that the single samples do not skip a significant section of the path.

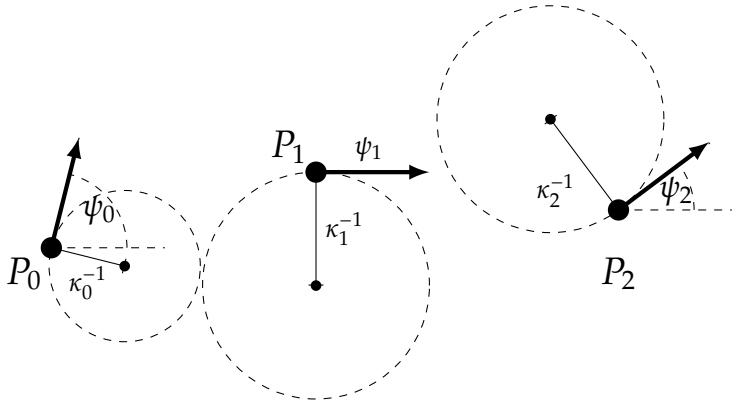


**Figure 9.** Exemplary G2 Hermite waypoint data, consisting of position $P$, orientation $\psi$ and curvature $\kappa$.

Alternatively a path planning algorithm may provide the reference path in form of an analytic curve expression. The most commonly used analytic curve expressions are parametric curves. In Appendix A, an overview on the basics of parametric curves and commonly used parametric curves expressions in path planning are given.

The more accurate the shape of a single analytic curve shall be specified, the more complex representations (e.g., high-order polynomials) are required. In practice, this yields an increased computational complexity and corresponding numerical issues. Hence, it is far more practicable to represent the planned path not as one single analytic curve but as a sequence of aligned curve segments, in which each are defined by simple analytic curves, yielding a so-called spline. The application of a segmented path needs an adaption of the elementary path operations stated in Section 4. The determination of a tracking control error has to be managed in two stages:

1. Global: Identification for the respective path segment.
2. Local: Application of the actual error computation within the path segment (see Section 4).

The identification of the respective path segment can inflate to a complex issue, since the actual optimal solution of this task requires the application of the path operation (step 2) on each path segment. In order to reduce the computational effort for step 1, a practicable workaround is to aim for a sub-optimal solution. Such a solution can be obtained, for example by applying the path operation to a simplified version of the path, like for example a linear interpolation of the way point within the path segments. However, the discrepancy between the sub-optimal solution and the optimal one, may become significant especially for reference paths including sharp turns, loops or cusps.

In summary, the path representations provided by state-of-the-art path planning algorithms feature a huge diversity. There is no obvious link for a potential generalization similar to the error classification for path tracking. However, according to Section 5, the

implementation of identified elementary path operations depends on the reference path representation. Therefore, it is reasonable to include an internal reference path representation into the interface, which, in general, may differ from a possible parametric path provided by the planner. Since it is straight forward to generate Hermite path data from a given parametric path, by path sampling, the generalized input format is defined as Hermite path data. The drawback of this definition is that the proposed interface may withdraw analytic curve expressions potentially provided by high-sophisticated planning components.

The choice for an internal path representation is a degree of freedom in the interface design. Appendix A gives important hints for the pros and cons of different curves. A common choice are polynomial splines. Such splines may be defined in different polynomial bases (e.g., monomial and Bernstein basis). The missing step to finalizing the wanted generic motion planning and control interface is the parametrization of the internal path with respect to given Hermite way point data. This is covered by the well investigate research field of interpolation. For sake of completeness, Appendix B surveys some state-of-the-art algorithms for the interpolation of Bezier splines, which serve as internal path representation for the exemplary implementation of the interface (cf. Section 8).

Figure 10 illustrates the intermediate result with respect to the input structure of the proposed generic interface.

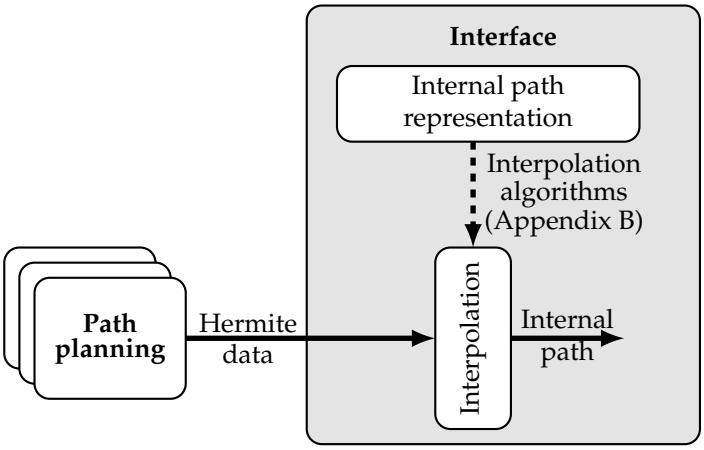


**Figure 10.** Interface concept to meet the input requirements defined by state-of-the-art reference path representations.

## 7. A Generic Path Planning and Tracking Interface

Combining the above considerations from path planning and path tracking side, the concept for the generic interface design shall now be summarized (see Figure 11). The three major steps in the implementation of such an interface are:

- Decision for an internal path representation.
- Implementation of corresponding Hermite waypoint data interpolation algorithms (see Appendix B), in order to accomplish input (planning) modularity with respect to different data types (G0, G1, ⋯).
- Implementation of corresponding error computation, based on the path operations discussed in Section 4 (path-line intersection, path-circle intersection and point projection), in order to accomplish output modularity (control).

The key design decision is the choice of an internal path representation. The internal path representation, on the one hand, has to be able to accurately describe a planned reference path. On the other hand the computational aspects of the required interpolation and path operation algorithms have to be considered. It is an important fact that this design decision impacts the final performance of the motion planning and control system, in terms of tracking performance (cf. Section 8) and computational effort.

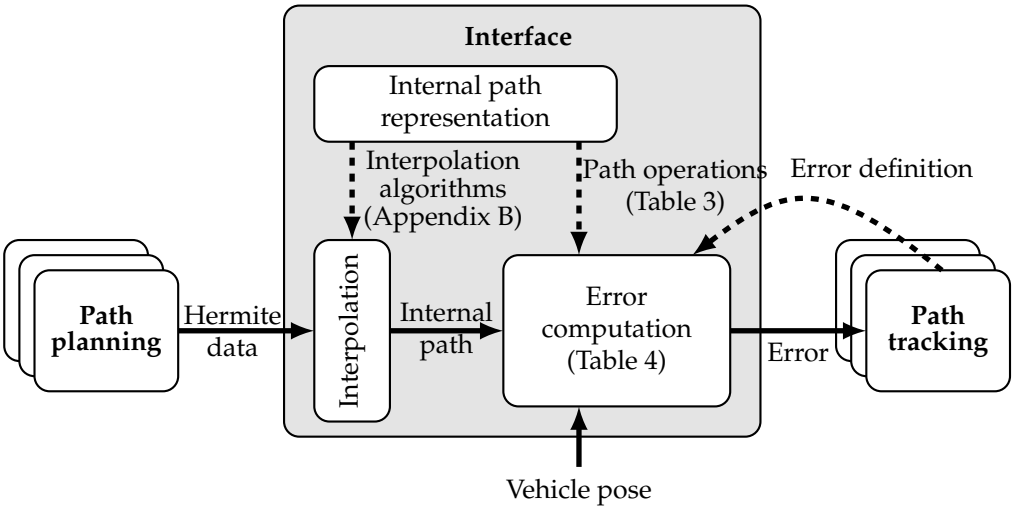


**Figure 11.** Interface for modular motion planning and control systems.

Such an interface is able to connect state-of-the-art path planning algorithms, which provide Hermite waypoint data, with state-of-the-art path tracking controller, which requires a lateral tracking error definition that is covered by the introduced error classifiers: vehicle reference, look-ahead and error orientation. Consequently, it enables the combination of a huge range of planning or tracking components without necessary interface modifications, which offers significant benefits in both simulation and operation:

- In simulation, it supports a straight-forward identification of an appropriate component set, based on iterative combination, simulation and evaluation, considering specific scenarios and corresponding KPIs. This is an important aspect of ODD-based AD function assembly. Furthermore, a specific error definition can be applied as common evaluation measure for a set of tracking controllers, which could not be compared on a quantitatively based on their different native error definition on a fair basis (cf. the example in Section 8).
- In operation, it enables the simultaneous execution of different software components as well as switching between different components. This on the one hand supports the design of AD-functions, for a set of varying ODDs extending the application range of Level 4 driving functions. On the other hand, it supports the application of redundant and fail-operational software in order to increase safety of an AD function.

In addition to these major benefits in simulation and operation, the application of the proposed interface offers additional possibilities in the AD function development. The resolution of the component interdependencies, which might impact the overall system performance, is a significant advantage in safety critical applications, like collision avoidance. Furthermore, as already mentioned in Sections 2 and 3.4, the tracking error computation can be executed also on a set of vehicle poses, generated by some motion prediction in order to support, e.g., MPC-based path tracking approaches. However, there might occur cases, which exclude the possibility of outsourcing the error computation to an interface component for some algorithmic reasons. In this case, the proposed interface still can operate as a reference path pre-processing unit, which may provide reference path sections of fixed quality, with respect to configurable requirements, e.g., equidistant sampling and path continuity, without any modifications of the planning components. Finally, the application of specific error definitions based on the current and predicted vehicle poses are also applicable and beneficial as risk indicators in threat and risk assessment (see for example [53,54]).

## 8. Exemplary Interface Application

In this final section, some of the aforementioned benefits and aspects of a generic interface shall be demonstrated in a practical use case. Starting point is a simplified but exemplary ODD-based AD function assembly problem. A path tracking controller shall be selected for application in an AD function for highway driving. This ODD arises a considerable number of specific scenarios. This example is restricted to an exemplary lane-change maneuver. Therefore, a standardized maneuver—the second half of a standardized double lane-change maneuver in [31]—is considered. The maneuver is performed at a constant speed of $v$ = 72 km/h. Since no obstacles have to be considered in this maneuver a simple path planner is used. It uses cubic Bezier splines (cf. Appendix A) to plan a smooth trajectory within the defined maneuver corridor. The planned path is provided to the generic interface as G2 Hermite data, with a sample distance of 5 m. In order to show the impact of the internal path representation on the control performance, two different interface implementations are applied in simulation. The first one uses a simple linear path representation. The provided Hermite path data are interpolated linearly, resulting in piece-wise straight sections, respectively piece-wise linear orientation and curvature. Note that the linear interpolation of orientation and curvature do not represent the actual orientation and curvature of the resulting polygonal path, which would be piece-wise constant respectively zero, but serves as an improved approximation. The second interface implementation is based on piece-wise polynomial path sections an applies the interpolation algorithms described in Appendix B. The G2 Hermite waypoint data are interpolated with quintic Bezier splines in order to achieve a G2 continuous internal reference path.

An exemplary set of two state-of-the-art path tracking controllers is used with a fixed parametrization (see Tables 5 and 6), including the applied lateral tracking error definition according to the proposed classification in Section 3:

- Stanley ([25]):

$$\delta(t) = e_\psi - k_{\text{ag}} \times v^2 \times \kappa_{ref} \times \sin e_\psi + \arctan\left(\frac{k \times e_{\text{lat}}}{k_{\text{soft}} + v}\right) \tag{13}$$

- PurePursuit ([41]):

$$\delta(t) = \arctan\left(\frac{2 \times l \times e_{\text{lat}}}{d_{\text{lh}}^2}\right) \tag{14}$$

**Table 5.** Parametrization of Stanley tracking controller.

| Parameter | Value |
|---|---|
| $k$ | 19 |
| $k_{\text{soft}}$ | 1 |
| $k_{\text{ag}}$ | 0.013 |
| vehicle reference | front |
| look-ahead | no |
| error orientation | path |

The two controllers shall just serve as an exemplary set of components for this demonstration. The performed evaluation can be extended straight forwardly for example to the entire set of tracking controllers listed in Table 2.

In order to compare the performance of different controllers, which, in general, apply different error definition, an additional lateral tracking error definition (vehicle reference: center of gravity, look-ahead: no, error orientation: heading) is used to provide a comparable error measure. The path planning system, the two exemplary interface implementations as well as the exemplary tracking controllers have been implemented in a Python-based

software framework. This framework has been used to perform the simulations, which are discussed within the following section.

**Table 6.** Parametrization of PurePursuit tracking controller.

| Parameter | Value |
|---|---|
| $l$ | 2.68 m |
| $d_{\mathrm{lh}}$ | 10 m |
| vehicle reference | rear |
| look-ahead | path |
| error orientation | heading |

### *8.1. Discussion*

Figure 12 shows the results of a series of four simulations (all combinations of the two controllers and the two interfaces implementations). According to the top plots, both controllers in principle accomplish the required scenarios task and keep the vehicle inside the defined scenario corridor. Due to its look-ahead the lateral error of the PurePursuit controller increase earlier, resulting in an earlier initiation of the lane-change maneuver. Furthermore, this yields a more smooth maneuver with reduced steering effort due to the damping behavior of a look-ahead in path tracking (cf. Section 3.2 and [23]). The drawback of this property is the curve-cutting behavior of the final vehicle motion, which brings the vehicle to the borders of the scenario corridor several times. Considering the defined common error measure, this fact substantiates in a considerable $\approx 1$ m) lateral offset of the vehicle's center of gravity with respect to the reference path. From this point of view, the Stanley controller shows superior tracking performance to the cost of an increased steering effort.

The comparison of the two different applied interfaces reveals the important fact that the internal path representation impacts the final tracking performance. The non-smooth linear reference path yields a non-smooth tracking error. The damping characteristic of the PurePursuit's look-ahead still ensures a satisfying steering command, which is almost equivalent to the corresponding steering command when applying the polynomial internal path. Contrary, the Stanley controller is affected through-out by this effect, yielding an unsatisfying jerky steering command. This consideration shows that in absence of an interface the Stanley controller is not applicable at all in this scenario in combination with the used planning system. Obviously, this is an illustrative edge-case example and the effect may be reduced when using a more dense sampling of the reference path, but, in general, this might be a fixed parameter of the planning system. The slight steering into the opposite direction of the Stanley controller results from an undesired property of the applied path interpolation algorithm, the so-called Runge's phenomenon.

In order to conclude this exemplary evaluation, based on the defined error measure lateral offset of the vehicle's center of gravity, the Stanley controller has to be favored in this scenario.

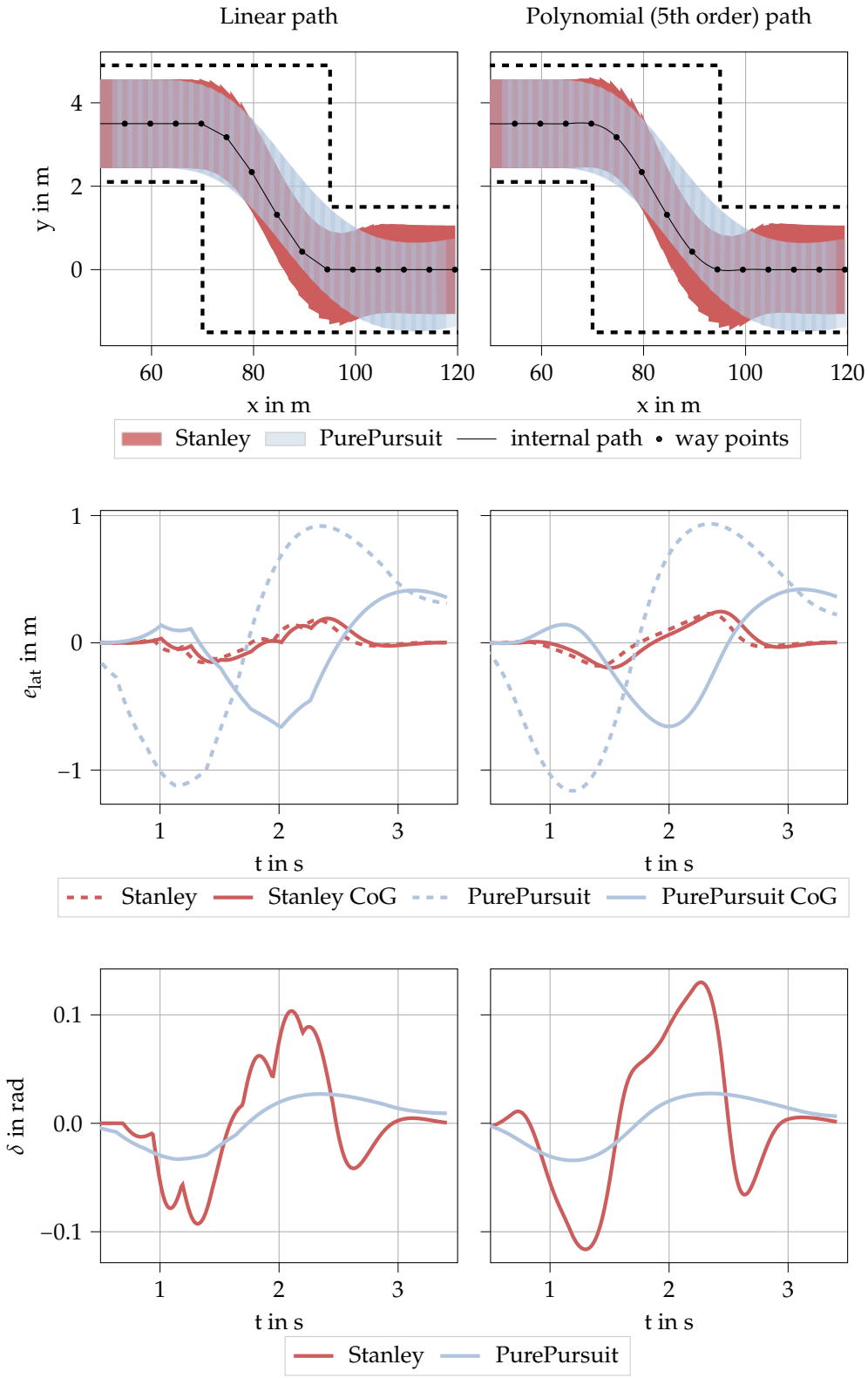


**Figure 12.** Simulation results: Lane-change maneuver [31] performed with two different path tracking controllers (Stanley [25] and PurePursuit [41]) using two different interface implementations (linear, respectively 5th order polynomial internal path representation). The plots on the top show the vehicle position, the plots in the middle illustrate the occurring lateral tracking errors (controller specific lateral tracking error + common error measure) and the bottom plots show the resulting steering commands computed by the two controllers.

## 9. Conclusions

Level 4 autonomous driving requires AD-functions, which are carefully matched to the specific ODD. The scientific state-of-the-art provides a tremendous amount of dedicated algorithms, which may be applicable for specific tasks of an AD-function in different ODDs. Hence, contrary to developing AD-functions from scratch for each ODD, it is beneficial to aim for a modular system architecture with generic interfaces, enabling fast combination and evaluation of sets of algorithms, without any code adaptions. This publication is dedicated to the design of such a generic interface between a local path planning and path tracking system. In order to ensure modularity with respect to different state-of-the-art path tracking controller, it contributes a classification of controllers based on the applied lateral tracking error definitions. Based on this classification the actual requirements from control side in the error computation are identified in terms of elementary path operations. This substantiates the advantage of including an internal reference path representation into the interface in order to resolve the interdependencies between the path planning and the path tracking system and, hence, to achieve the required input modularity (on planning side) and output modularity (on control side). With these building blocks, the publication contributes a generic interface concept and a solid theoretical basis for occurring design decisions in the implementation. Two exemplary implementations are finally applied in a demonstrative ODD-based AD assembly task. The application of such an interface offers significant advantages in simulation, like straight-forward combination, evaluation, and benchmarking of set of components in different scenarios, as well as in operation, as a key element for fail-operational and ODD-adaptive AD-function design. A concluding overview on the technical content of the publication is represented in Table 7.

**Table 7.** Overview on the proposed generic interface concept for path planning and tracking.

| | |
|---|---|
| **Approach** | • Analysis of requirements based on state-of-the-art components<br>• Identification of potentials for generalization<br>• Definition of component independent interface tasks and requirements |
| **Concept** | • Interface internal continuous reference path representation (Section 6)<br>• Generalization of tracking error definition based on classifiers ((11) and Section 3)<br>• Implementation of elementary path operations (Table 3) based on the internal path representation (Section 4)<br>• Generalized tracking error computation based on error definition and path operations (Table 4)<br>• Parametrization of the internal path by Hermite interpolation of reference path data (Appendix B) |
| **Benefits** | • Arbitrary combination of state-of-the-art path planning and tracking algorithms<br>• Avoidance of component independent input/output data processing within the components<br>• Push modular ODD-based AD function design (iterative combination, simulation and evaluation of path planning and tracking components)<br>• Support application of redundant and fail-operational software design to increase safety of an AD function (redundant operation and ODD-based switching of components) |

**Author Contributions:** Conceptualization, J.R. and M.S.; methodology, J.R.; software, J.R.; validation, J.R. and M.S.; formal analysis, J.R. and M.S.; investigation, J.R.; writing—original draft preparation, J.R.; writing—review and editing, J.R., M.S. and D.W.; supervision, M.S. and D.W.; funding acquisition, D.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** The authors would like to acknowledge the financial support within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research

Promotion Agency (FFG) has been authorized for the program management. Supported by TU Graz Open Access Publishing Fund.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. Parametric Path

This section gives a theoretical basis on parametric path representation and exemplary concepts.

### *Appendix A.1. Basics*

Analytic curve expressions can be given in terms of a implicit, explicit or parametric expression:

$$\text{implicit: } f(x,y) = 0, \quad \text{explicit: } y = f(x), \quad \text{parametric: } \gamma(\tau) = \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^T. \tag{A1}$$

In general, a parametric expression is most convenient, since it enables a global definition of arbitrary paths (from a starting point $P_\mathrm{a}$ to a target point $P_\mathrm{b}$),

$$\gamma(\tau) = \begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix}: \quad \gamma(\tau_\mathrm{a}) = \begin{bmatrix} x(\tau_\mathrm{a}) \\ y(\tau_\mathrm{a}) \end{bmatrix} = P_\mathrm{a} \quad \text{and} \quad \gamma(\tau_\mathrm{b}) = \begin{bmatrix} x(\tau_\mathrm{b}) \\ y(\tau_\mathrm{b}) \end{bmatrix} = P_\mathrm{b}, \tag{A2}$$

without any bijection issues as they occur for implicit and explicit expressions and which require path sectioning and transformation to local coordinate systems.

The curve parameter $\tau$ is a monotonically increasing parameter, related to the progress along the path. It is a degree of freedom and can be used for normalization,

$$\tau_\mathrm{a} \stackrel{!}{=} 0, \quad \tau_\mathrm{b} \stackrel{!}{=} 1, \tag{A3}$$

or to achieve a path length parametrization $\tau = s$ (also known as natural, arc-length or chord-length parametrization):

$$\frac{\partial s}{\partial \tau} = \sqrt{x'^2 + y'^2} \stackrel{!}{=} 1, \tag{A4}$$

holds. $x'$ and $y'$ are the derivatives with respect to the curve parameter $\frac{\partial x}{\partial \tau}$ and $\frac{\partial y}{\partial \tau}$. If the curve parameter in this case is interpreted as time ($\tau = t$), the curve is passed with a speed of $v = 1$ m/s. Therefore, this parametrization is also called unit-speed parametrization (see for example [55]). Orientation and curvature of a parametric path compute as:

$$\theta(\tau) = \arctan\left(\frac{y'}{x'}\right), \tag{A5}$$

$$\kappa(\tau) = \frac{\partial \theta}{\partial s} = \frac{\partial \theta}{\partial \tau}\frac{\partial \tau}{\partial s} = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{\frac{3}{2}}}, \tag{A6}$$

### *Appendix A.2. Clothoidal Path*

A widely used curve in natural parametrization is the clothoid or Euler spiral. It is defined by a linear curvature with respect to arc length,

$$\kappa(s) = \sigma \times s, \tag{A7}$$

and is widely used as transition curve between straight and circular road segments in road network design to offer a good steering behavior. Therefore, for it is an important parametric curve candidate for path planning (see, for example, [5,56–59]). The consideration of clothoid $x, y$-coordinates by integration with respect to (A5) and (A6),

$$x(s) = \int_0^s \cos\left(\frac{\sigma\tau^2}{2}\right)\mathrm{d}\tau, \quad y(s) = \int_0^s \sin\left(\frac{\sigma\tau^2}{2}\right)\mathrm{d}\tau, \tag{A8}$$

reveals the main drawback of a natural curve parametrization: Whereas arc length-based curve parameters (curvature, ...) feature a simple form, the functions for the coordinates may be transcendental functions (in this case Fresnel-Integrals), which cannot be solved analytically. This property of curves in natural parametrization complicates a coordinate-based definition, like curve fitting or interpolation, which is much more straight forward for other curve parametrizations. Therefore, several approaches have been published, aiming at approximation of clothoid section with other parametric curves (see, for example, [60–62]) to overcome this drawback while maintaining the advantage of a bounded path curvature.

*Appendix A.3. Polynomial Path*

A widely-used and, hence, important parametric path is the polynomial path. It is defined by a set of coefficients,

$$\mathbf{a}_{\mathrm{x}}^{T} = \begin{bmatrix} a_{\mathrm{x},0} & \dots & a_{\mathrm{x},n} \end{bmatrix}, \quad \mathbf{a}_{\mathrm{y}}^{T} = \begin{bmatrix} a_{\mathrm{y},0} & \dots & a_{\mathrm{y},n} \end{bmatrix}, \tag{A9}$$

with respect to some set of polynomial basis functions—monomial basis functions,

$$\boldsymbol{\tau}_{\mathrm{m},n}^{T} = \begin{bmatrix} 1 & \tau & \dots & \tau^{n} \end{bmatrix}, \tag{A10}$$

in the most general case:

$$\gamma(\tau) = \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^{T} = \begin{bmatrix} \mathbf{a}_{\mathrm{x}}^{T}\boldsymbol{\tau}_{\mathrm{m},n} & \mathbf{a}_{\mathrm{y}}^{T}\boldsymbol{\tau}_{\mathrm{m},n} \end{bmatrix}^{T}. \tag{A11}$$

Polynomials in monomial basis can be efficiently evaluated using *Horner's scheme* [63], which features a minimum number of additions and multiplications. Due to possibly huge differences in the range of the single polynomial coefficients, however, numerical instabilities might occur. The Bernstein basis is a widely used alternative to the monomial basis. For a polynomial of order $n$, the basis functions,

$$\boldsymbol{\tau}_{\mathrm{b},n}^{T} = \begin{bmatrix} \tau_{\mathrm{b},n,0} & \tau_{\mathrm{b},n,1} & \dots & \tau_{\mathrm{b},n,n} \end{bmatrix}, \tag{A12}$$

compute as:

$$\tau_{\mathrm{b},n,i} = \binom{n}{i}\tau^{i}(1-\tau)^{n-i} \quad \text{with:} \quad i = [0, 1, \dots, n], \tag{A13}$$

where the relation

$$\sum_{i=0}^{n} \tau_{\mathrm{b}n,i}(\tau) \equiv 1 \quad \text{and:} \quad \tau \in (0,1), \tag{A14}$$

holds. A polynomial curve in Bernstein basis is known as Bezier curve:

$$\gamma(\tau) = \begin{bmatrix} x(\tau) & y(\tau) \end{bmatrix}^{T} = \begin{bmatrix} \mathbf{b}_{\mathrm{x}}^{T}\boldsymbol{\tau}_{\mathrm{b},n} & \mathbf{b}_{\mathrm{y}}^{T}\boldsymbol{\tau}_{\mathrm{b},n} \end{bmatrix}^{T}. \tag{A15}$$

The $(n+1)$ pairs of $x, y$-coefficients,

$$\begin{bmatrix} B_0 & \dots & B_n \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathrm{x}}^{T} \\ \mathbf{b}_{\mathrm{y}}^{T} \end{bmatrix}, \tag{A16}$$

the so-called control points, form a convex hull to the curve (see Figure A1) and, hence, offer a comprehensible geometric interpretation of the polynomial coefficients:

$$\gamma(\tau) = \begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix} = \sum_{i=0}^{n} B_i \times \tau_{\mathrm{b},n,i}(t). \tag{A17}$$

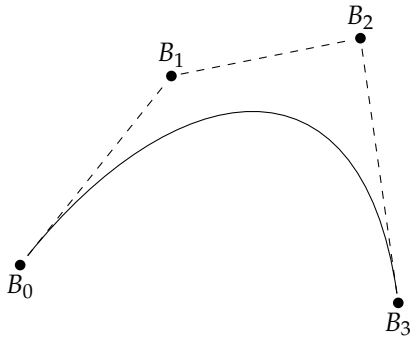


**Figure A1.** Exemplary cubic Bezier curve from $B_0$ to $B_3$ controlled by control $B_1$ and $B_2$.

Bezier curves play an important role in computer graphics, and also in path planning. With the *De Casteljau's algorithm* there exists a strong algorithm to evaluate a Bezier curve. It is not as efficient as the *Horner's scheme* but more numerically stable. As also the Bernstein basis finally consists of monomial terms, there exists a static transformation between these two bases:

$$\begin{bmatrix} A_{n,0}^T \\ \vdots \\ A_{n,n}^T \end{bmatrix} = \mathbf{M}_n \begin{bmatrix} B_{n,0}^T \\ \vdots \\ B_{n,n}^T \end{bmatrix} \tag{A18}$$

This transformation may be used to adaptively utilize the advantages of the different bases. Without proof the transformation matrix $\mathbf{M}_n$ can be computed with the following Hadamard product:

$$\mathbf{M}_n = \mathbf{Q}_n \circ \mathbf{P}_{\mathrm{L},-n}. \tag{A19}$$

With a polynomial coefficient matrix $\mathbf{Q}_n$ with switching signs with elements $q_{i,j}$:

$$q_{i,j} := \binom{n}{i}, \quad \text{for:} \quad i,j \in [0,\ldots,n] \tag{A20}$$

and lower triangular pascal matrix extension to negative coefficients $\mathbf{P}_{\mathrm{L},-n}$, which is for example,

$$\mathbf{P}_{\mathrm{L},-3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 1 & -3 & 3 & -1 \end{bmatrix}, \tag{A21}$$

for $n = 3$. This relation offers a simple approach to assemble a transformation matrix for specific order of the polynomials, which has not been published yet to the knowledge of the authors.

In order to improve the quality of a polynomial approximations of specific curve sections, e.g., conic sections, while maintaining a low polynomial degree, there exists several generalizations (see, for example, [64]) like rational polynomial functions, B-splines and there combination: non-uniform rational B-splines (NURBS).

## Appendix B. Hermite Data Interpolation

Path interpolation is an extensively researched topic and is considered as a special case of path smoothing (see [65]). In [65], a comprehensive overview on the state-of-the-art in path smoothing and interpolation is given. In [66], the mathematical properties of different interpolation functions are considered in very detail. This section shall not survey the complete state-of-the-art in path interpolation, but assemble basic definitions and a survey on interpolation algorithms for Bezier splines. The task of interpolation is to find a

parametric curve expression for each path segment, which matches the given G$n$ Hermite data (see Figure A2), with $n = 0, 1, \cdots$.

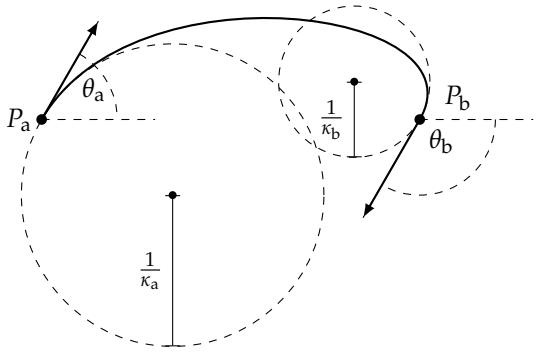


**Figure A2.** G2 Hermite data interpolation.

The number of matched geometric path information determine the minimal geometric continuity of the total path at the waypoints, i.e., the transition points between the single spline segments. If the path, furthermore, features continuous orientation (and curvature, ...) on the entire definition domain, the curve is called to have G1 (G2, ...) continuity. In contrast to this so-called geometric continuity, differential continuity refers to the derivatives of the curve with respect to the curve parameter $\tau$. Differential continuity always implies geometric continuity. If the curve parameter $\tau$ is not applied also for longitudinal velocity planning, differential continuity is a too restrictive requirement, which does not provide any additional geometric features of the curve. However, in practice it is often more convenient to require differential continuity as it will be shown on a comprehensive example of a G2 Hermite interpolation with Bezier curves.

Consider a set of G2 Hermite way point data in two points $P_a$ and $P_b$. This gives eight boundary conditions and, hence, a cubic Bezier spline (determined by four way points) should be sufficient to meet these conditions:

$$x_a, y_a, \theta_a, \kappa_a, x_b, y_b, \theta_b, \kappa_b \mapsto B_0, B_1, B_2, B_3, \tag{A22}$$

Due to non-linearity of tangent direction and curvature of a general parametric curve, the analysis of the existence and uniqueness of the solutions to this interpolation problem is rather complex. In fact, a cubic spline is not sufficient to interpolate arbitrary G2 Hermite data. Ref. [67] gives a proof that the necessary order for Hermite interpolation of arbitrary G2 data with a Bezier curves is $3 \leq n \leq 5$ depending on the given boundary conditions for tangent directions and curvatures. In [68], conditions for the G2 boundary conditions on the existence and uniqueness of a cubic curve solution are given, restricting on shape preserving curves. In summary, the Hermite interpolation of arbitrary G2 data with Bezier curves of fixed order, in general, requires quintic curves. According to [67], this, consequently, means that occasionally up to 4 additional degrees of freedom occur.

An alternative approach is to implicitly solve the problem, requiring differential continuity (C2 for this example). This enlarges the original set of eight boundary conditions to twelve,

$$x_a, y_a, x'_a, y'_a, x''_a, y''_a, x_b, y_b, x'_b, y'_b, x''_b, y''_b \mapsto B_0, B_1, B_2, B_3, B_4, B_5 \tag{A23}$$

and, consequently, directly requires a quintic Bezier curve (six control points). In [69,70], a parameter vector $\boldsymbol{\eta}$ has been proposed to utilize the additional degrees of freedom compared to the original problem for later curvature minimization. An extension of this concept for G3 Hermite interpolation has been published in [71,72].

A different common interpolation problem is that a specific geometric continuity between the single spline segments of a curve is required, but not (or not all of them) fixed to a given value. Following again an implicit approach over the differential continuity, a linear equation system can be set up to determine the coefficients of the curve segments. In

order to obtain a fully determined equation system, additional global boundary conditions (initial conditions for the first segment and final conditions for the last segment) have to be specified. The necessary spline order $n$ of the curve segments and the resulting number of additional global boundary conditions $n_{\mathrm{BC}}$ can be determined straight forwardly considering C$x$ input data at $N$ way points and required C$y$ path continuity (with $x \leq y$):

$$\underbrace{2}_{2\mathrm{D}} \cdot \overbrace{2(x+1)}^{\text{fixed local BCs}} \underbrace{(N-1)}_{\text{splines}} + \overbrace{2}^{2\mathrm{D}} \underbrace{(y-x)}_{\text{free lokal BCs}} \overbrace{(N-2)}^{\text{joints}} + \underbrace{n_{\mathrm{BC}}}_{\text{global BCs}} \equiv \overbrace{2}^{2\mathrm{D}} \underbrace{(N-1)}_{\text{splines}} \overbrace{(n+1)}^{\text{coeffs./spline}} \tag{A24}$$

The evaluation of this identity in $N$ (the spline order should be independent of the number of way points) yields:

$$n_{\mathrm{BC}} = 2(y-x) \quad \text{and} \quad n = x + y + 1. \tag{A25}$$

Table A1 lists the evaluation of (A25). These simple relations are quite useful in order to implement input data adaptive interpolation algorithm, but have not been published yet to the knowledge of the authors.

In the special case of $x = 0$ and $y$ being an odd number and a homogeneous choice of the global boundary conditions, the resulting splines are called natural splines (see [66]). Such natural splines are characterized by the possibility to continuously extend them with polynomials of order $\frac{n-1}{2}$ outside of the nominal definition space, i.e., before the starting point $P_0$ respectively after the end point $P_{\mathrm{N}-1}$.

While the splines, as proposed above, are simple to compute, they suffer from several drawbacks like an asymmetric shape in the case that the number of global boundary conditions is not a multiple of four. Furthermore, they are global splines, which means that appending additional way point data affects the entire spline. Another drawback of, in general, all polynomial approaches, is that, although a specific geometric or differential continuity is ensured at the spline transitions, there are, in general, no guarantees about the continuity within the single spline segments. In fact, for spline order $n \geq 3$ cusps (curvature discontinuities) may occur. [73] gives necessary and sufficient conditions for the existence of such cusps. To handle this problem, special types of more restrictive polynomials, which result in so-called regular curves (curves without cusps) have been proposed. For example, in [74] Bezier spirals (a spiral is a curve with monotonic curvature, i.e., without interior curvature extrema) are used to ensure specific continuity.

**Table A1.** Necessary spline order and number of free boundary conditions for given Hermite data and required differential path continuity.

| Data | Continuity | Spline Order | Boundary Conditions |
|---|---|---|---|
| C0 | C2 | 3 | 4 |
| C1 | C2 | 4 | 2 |
| C0 | C3 | 4 | 6 |
| C1 | C3 | 5 | 4 |

## References


1. SAE. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*; SAE: Warrendale, PA, USA, 2018.
2. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
3. Ulbrich, S.; Reschka, A.; Rieken, J.; Ernst, S.; Bagschik, G.; Dierkes, F.; Nolte, M.; Maurer, M. Towards a functional system architecture for automated vehicles. *arXiv* **2017**, arXiv:1703.08557.
4. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2007.

5. Paden, B.; Cap, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
6. Sorniotti, A.; Barber, P.; De Pinto, S. Path Tracking for Automated Driving: A Tutorial on Control System Formulations and Ongoing Research. In *Automated Driving*; Watzenig, D., Horn, M., Eds.; Springer: Cham, Switzerland, 2017; Chapter 5, pp. 71–140.
7. You, D.; Wang, H.; Yang, K. State-of-the-art and trends of autonomous driving technology. In Proceedings of the TEMS-ISIE 2018-1st Annual International Symposium on Innovation and Entrepreneurship of the IEEE Technology and Engineering Management Society, Beijing, China, 30 March–1 April 2018. [CrossRef]
8. Kato, S.; Tokunaga, S.; Maruyama, Y.; Maeda, S.; Hirabayashi, M.; Kitsukawa, Y.; Monrroy, A.; Ando, T.; Fujii, Y.; Azumi, T. Autoware on board: Enabling autonomous vehicles with embedded systems. In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), Porto, Portugal, 11–13 April 2018; pp. 287–296.
9. Hamid, U.Z.A.; Saito, Y.; Rahman, M.A.A.; Zamzuri, H.; Raksincharoensak, P. A review on threat assessment, path planning and path tracking strategies for collision avoidance systems of autonomous vehicles. *Int. J. Veh. Auton. Syst.* **2018**, *14*, 134. [CrossRef]
10. Tang, J. Review: Analysis and Improvement of Traffic Alert and Collision Avoidance System. *IEEE Access* **2017**, *5*, 21419–21429. [CrossRef]
11. Werling, M.; Gröll, L. From flatness-based trajectory tracking to path following. In Proceedings of the 2009 IEEE Intelligent Vehicles Symposium, Xi'an, China, 3–5 June 2009; pp. 271–1275. [CrossRef]
12. De Luca, A.; Oriolo, G.; Samson, C. Feedback control of a nonholonomic car-like robot. In *Robot Motion Planning and Control*; Laumond, J.-P., Ed.; Lecture Notes in Control and Information Sciences; Springer: Berlin/Heidelberg, Germany, 1998; Volume 229, Chapter 4, pp. 171–253.
13. Snider, J.M. Automatic Steering Methods for Autonomous Automobile Path Tracking. In *Technical Report CMU-RI-TR-09-08*; Robotics Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2009.
14. Chatzikomis, C.; Sorniotti, A.; Gruber, P.; Zanchetta, M.; Willans, D.; Balcombe, B. Comparison of Path Tracking and Torque-Vectoring Controllers for Autonomous Electric Vehicles. *IEEE Trans. Intell. Veh.* **2018**, *3*, 559–570. [CrossRef]
15. Calzolari, D.; Schurmann, B.; Althoff, M. Comparison of trajectory tracking controllers for autonomous vehicles. In Proceedings of the IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, Yokohama, Japan, 16–19 October 2018; pp. 1–8. [CrossRef]
16. Rouchon, P.; Fliess, M.; Levine, J.; Martin, P. Flatness, motion planning and trailer systems. In Proceedings of the 32nd Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993; pp. 2700–2705.
17. Kritayakirana, K.; Gerdes, J.C. Using the centre of percussion to design a steering controller for an autonomous race car. *Veh. Syst. Dyn.* **2012**, *50*, 33–51. [CrossRef]
18. Peters, S.C.; Frazzoli, E.; Iagnemma, K. Differential flatness of a front-steered vehicle with tire force control. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011, pp. 298–304. [CrossRef]
19. Xu, S.; Peng, H. Design, Analysis, and Experiments of Preview Path Tracking Control for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 48–58. [CrossRef]
20. Nestlinger, G.; Stolz, M. Bumpless Transfer for Convenient Lateral Car Control Handover. *IFAC-PapersOnLine* **2016**, *49*, 132–138. [CrossRef]
21. Broggi, A.; Bertozzi, M.; Fascioli, A.; Bianco, C.G.L.; Piazzi, A. The ARGO Autonomous Vehicle's Vision and Control Systems. *Int. J. Intell. Control Syst.* **1999**, *3*, 409–441.
22. Yuan, X.; Huang, G.; Shi, K. Improved Adaptive Path Following Control System for Autonomous Vehicle in Different Velocities. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3247–3256. [CrossRef]
23. Guldner, J.; Sienel, W.; Tan, H.S.; Ackermann, J.; Patwardhan, S.; Bünte, T. Robust automatic steering control for look-down reference systems with front and rear sensors. *IEEE Trans. Control Syst. Technol.* **1999**, *7*, 2–11. [CrossRef]
24. Hu, C.; Wang, R.; Yan, F.; Chen, N. Should the Desired Heading in Path Following of Autonomous Vehicles be the Tangent Direction of the Desired Path? *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 3084–3094. [CrossRef]
25. Hoffmann, G.M.; Tomlin, C.J.; Montemerlo, M.; Thrun, S. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In Proceedings of the American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 2296–2301. [CrossRef]
26. Kolb, J.K.; Nitzsche, G.; Wagner, S. A simple yet efficient Path Tracking Controller for Autonomous Trucks. *IFAC-PapersOnLine* **2019**, *52*, 307–312. [CrossRef]
27. Sun, C.; Zhang, X.; Xi, L.; Tian, Y. Design of a path-tracking steering controller for autonomous vehicles. *Energies* **2018**, *11*, 1451. [CrossRef]
28. Sun, C.; Zhang, X.; Zhou, Q.; Tian, Y. A Model Predictive Controller with Switched Tracking Error for Autonomous Vehicle Path Tracking. *IEEE Access* **2019**, *7*, 53103–53114. [CrossRef]
29. Chen, C.; Jia, Y.; Shu, M.; Wang, Y. Hierarchical Adaptive Path-Tracking Control for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2900–2912. [CrossRef]
30. Hu, C.; Chen, Y.; Wang, J.; Member, S. Fuzzy Observer-Based Transitional Path-Tracking Control for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–11. [CrossRef]

31. Bruschetta, M.; Picotti, E.; Mion, E.; Chen, Y.; Beghi, A.; Minen, D. A Nonlinear Model Predictive Control based Virtual Driver for high performance driving. In Proceedings of the CCTA 2019-3rd IEEE Conference on Control Technology and Applications, Hong Kong, China, 19–21 August 2019; pp. 9–14. [CrossRef]
32. Zhang, C.; Hu, J.; Qiu, J.; Yang, W.; Sun, H.; Chen, Q. A Novel Fuzzy Observer-Based Steering Control Approach for Path Tracking in Autonomous Vehicles. *IEEE Trans. Fuzzy Syst.* **2019**, *27*, 278–290. [CrossRef]
33. Hiraoka, T.; Nishihara, O.; Kumamoto, H. Automatic path-tracking controller of a four-wheel steering vehicle. *Veh. Syst. Dyn.* **2009**, *47*, 1205–1227. [CrossRef]
34. Tieber, K. Motion planning and control of a people mover. Master's Thesis, Graz University of Technology, Graz, Austria, 2019.
35. Samson, C. Path Following And Time-Varying Feedback Stabilization of a Wheeled Mobile Robot. In Proceedings of the Second International Conference on Automation, Robotics and Computer Vision, Singapore, 16–18 September 1992; vol.3.
36. Dominguez, S.; Ali, A.; Garcia, G.; Martinet, P. Comparison of lateral controllers for autonomous vehicle : Experimental results. In Proceedings of the IEEE Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1418–1423. [CrossRef]
37. Solea, R.; Nunes, U. Trajectory planning with velocity planner for fully-automated passenger vehicles. In Proceedings of the IEEE Conference on Intelligent Transportation Systems (ITSC), Toronto, ON, Canada, 17–20 September 2006; pp. 474–480. [CrossRef]
38. Ackermann, J.; Guldner, J.; Sienel, W.; Steinhauser, R. Linear and Nonlinear Controller Design for Robust Automatic Steering. *IEEE Trans. Control Syst. Technol.* **1995**, *3*, 132–143. [CrossRef]
39. Elkaim, G.H.; Connors, J.; Nagle, J. The overbot: An off-road autonomous ground vehicle testbed. In Proceedings of the Institute of Navigation-19th International Technical Meeting of the Satellite Division (ION GNSS 2006), Fort Worth, TX, USA, 26–29 September 2006; Volume 3, pp. 1449–1456.
40. Sentouh, C.; Chevrel, P.; Mars, F.; Claveau, F. A sensorimotor driver model for steering control. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; pp. 2462–2467. [CrossRef]
41. Coulter, R.C. Implementation of the Pure Pursuit Path Tracking Algorithm. In *Technical Report CMU-RI-TR-92-01*; Carnegie Mellon University: Pittsburgh, PA, USA, 1992.
42. Cibooglu, M.; Karapinar, U.; Soylemez, M.T. Hybrid controller approach for an autonomous ground vehicle path tracking problem. In Proceedings of the 2017 25th Mediterranean Conference on Control and Automation (MED 2017), Valletta, Malta, 3–6 July 2017; Volume 8, pp. 583–588. [CrossRef]
43. Spencer, M.R. Polynomial Real Root Finding in Bernstein Form Polynomial Real Root Finding in Bernstein Form. Ph.D Thesis, Birgham Young University-Provo, Provo, UT, USA, 1994.
44. McNamee, J.M.; Pan, V.Y. Efficient polynomial root-refiners: A survey and new record efficiency estimates. *Comput. Math. Appl.* **2012**, *63*, 239–254. [CrossRef]
45. Meek, D.S.; Walton, D.J. A note on finding clothoids. *J. Comput. Appl. Math.* **2004**, *170*, 433–453. [CrossRef]
46. Ma, Y.L.; Hewitt, W.T. Point inversion and projection for NURBS curve: Control polygon approach. In Proceedings of the Theory and Practice of Computer Graphics (TPCG), Birmingham, UK, 5 June 2003; Volume 20, pp. 113–120. [CrossRef]
47. Chen, X.D.; Zhou, Y.; Shu, Z.; Su, H.; Paul, J.C.; Improved Algebraic Algorithm On Point Projection For Bézier Curves. In Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007), Iowa City, IA, USA, 13–15 August 2007.
48. Song, H.C.; Xu, X.; Shi, K.L.; Yong, J.H. Projecting points onto planar parametric curves by local biarc approximation. *Comput. Graph.* **2014**, *38*, 183–190. [CrossRef]
49. Hu, S.M.; Wallner, J. A second order algorithm for orthogonal projection onto curves and surfaces. *Comput. Aided Geom. Des.* **2005**, *22*, 251–260. [CrossRef]
50. Latombe, J.C. *Robot Motion Planning*; Springer: Boston, MA, USA, 1991.
51. Lavalle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
52. González, D.; Pérez, J.; Milanés, V.; Nashashibi, F. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
53. Li, Y.; Zheng, Y.; Morys, B.; Pan, S.; Wang, J.; Li, K. Threat Assessment Techniques in Intelligent Vehicles: A Comparative Survey. *IEEE Intell. Transp. Syst. Mag.* **2020**. [CrossRef]
54. Lefèvre, S.; Vasquez, D.; Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* **2014**, *1*, 1–14. [CrossRef]
55. Choe, R.; Puig-Navarro, J.; Cichella, V.; Xargay, E.; Hovakimyan, N. Trajectory Generation Using Spatial Pythagorean Hodograph Bézier Curves. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Kissimee, FL, USA, 5–9 January 2015; pp. 1–32. [CrossRef]
56. Scheuer, A.; Laugier, C. Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots. In Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190), Victoria, BC, Canada, 17 October 1998; Volume 1, pp. 25–31. doi: 10.1109/IROS.1998.724591. [CrossRef]
57. Gim, S.; Adouane, L.; Lee, S.; Dérutin, J.P. Clothoids Composition Method for Smooth Path Generation of Car-Like Vehicle Navigation. *J. Intell. Robot. Syst. Theory Appl.* **2017**, *88*, 129–146. [CrossRef]
58. Bertolazzi, E.; Frego, M. On the G 2 Hermite Interpolation Problem with clothoids. *J. Comput. Appl. Math.* **2018**, *341*, 99–116. [CrossRef]

59. Meek, D.S.; Walton, D.J. Clothoid Spline Transition Spirals. *Math. Comput.* **1992**, *59*, 117. [CrossRef]
60. Vázquez-Méndez, M.E.; Casal, G. Clothoid computation: A simple and efficient numerical algorithm. *J. Surv. Eng.* **2016**, *142*, 1–9, [CrossRef]
61. Chen, Y.; Cai, Y.; Zheng, J.; Thalmann, D. Accurate and Efficient Approximation of Clothoids Using Bézier Curves for Path Planning. *IEEE Trans. Robot.* **2017**, *33*, 1242–1247. [CrossRef]
62. Montés, N.; Herraez, A.; Armesto, L.; Tornero, J. Real-time clothoid approximation by rational bezier curves. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 2246–2251. [CrossRef]
63. Horner, W.G. A new method of solving numerical equations of all orders, by continuous approximation. *Philos. Trans. R. Soc. Lond.* **1819**, *109*, 308–335.
64. Piegl, L.; Tiller, W. *The NURBS Book*; Springer: Berlin/Heidelberg, Germany, 1995.
65. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors* **2018**, *18*, 3170. [CrossRef]
66. Bojanov, B.D. *Spline Functions and Multivariate Interpolations*; Springer Netherlands Imprint Springer: Dordrecht, The Newtherland, 1993.
67. Schaback, R. Optimal Geometric Hermite Interpolation of Curves. In *Mathematical Methods for Curves and Surfaces II*; Daehlen, M., Lyche, T., Schumaker, L.L., Eds.; Vanderbilt University Press: Nashville, TN, US, 1998; pp. 1–12.
68. Krajnc, M. Interpolation scheme for planar cubic G 2 spline curves. *Acta Appl. Math.* **2011**, *113*, 129–143. [CrossRef]
69. Piazzi, A.; Guarino Lo Bianco, C. Quintic G/sup 2/-splines for trajectory planning of autonomous vehicles. In Proceedins of the IEEE Intelligent Vehicles Symposium 2000, Dearborn, MI, USA, 3–5 October 2000; pp. 198–203. [CrossRef]
70. Piazzi, A.; Lo Bianco, C.G.; Bertozzi, M.; Fascioli, A.; Broggi, A. Quintic G2-Splines for the Iterative Steering of Vision-Based Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 27–36. [CrossRef]
71. Piazzi, A.; Romano, M.; Lo Bianco, C.G. G3-splines for the path planning of wheeled mobile robots. In Proceedings of the European Control Conference (ECC 2003), Cambridge, UK, 1–4 September 2003; pp. 1845–1850. [CrossRef]
72. Piazzi, A.; Lo Bianco, C.; Romano, M. n3-Splines for the Smooth Path Generation of Wheeled Mobile Robots. *IEEE Trans. Robot.* **2007**, *23*, 1089–1095. [CrossRef]
73. Manocha, D.; Canny, J.F. Detecting cusps and inflection points in curves. *Comput. Aided Geom. Des.* **1992**, *9*, 1–24. [CrossRef]
74. Walton, D.J.; Meek, D.S. Cubic Bezier Spiral Segments for Planar G2 Curve Design. In Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, Franschhoek, South Africa, 21–23 June 2010; pp. 21–26. [CrossRef]