IEEE Open Journal of
Intelligent Transportation
Systems

Received 2 December 2022; revised 6 January 2023; accepted 8 January 2023. Date of publication 11 January 2023; date of current version 27 January 2023.

Digital Object Identifier 10.1109/OJITS.2023.3235986

Anytime Tree-Based Trajectory Planning for Urban Driving

CHRISTOPH ZIEGLER AND JÜRGEN ADAMY

Department of Control Methods and Intelligent Systems, Technical University of Darmstadt, 64283 Darmstadt, Germany

CORRESPONDING AUTHOR: C. ZIEGLER (e-mail: christoph.ziegler@tu-darmstadt.de)

This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG—German Research Foundation) and in part by the Open Access Publishing Fund of Technical University of Darmstadt.

ABSTRACT The personal mobility of the future will be changed significantly by autonomous driving. To realize this vision, the complex task of trajectory planning needs to be solved. In this article, a novel planning concept, CarPre trajectory planning, based on Monte-Carlo tree search, is presented. Using a speed-dependent steering angle transformation, the state space of a kinematic single track model is discretized. The planner can then choose between different actions, each consisting of a discrete-value pair of an acceleration and a steering rate. With this, an equitemporal search tree is created to compute the future trajectory. Using Monte-Carlo simulations, the influence of short-term actions of the vehicle can be evaluated over a longer planning horizon. Thus, the current best solution can be accessed at any point during computation, enabling real-time applications. Furthermore, the discretized search tree enables easy checking of complex constraints dependent on binary or continuous variables. The concept is verified on a real test vehicle in a lane keeping maneuver. Through initial testing, a pleasant driving experience is perceived, which indicates future acceptance of the real-time capable algorithm.

INDEX TERMS Anytime, automated vehicles, MCTS, motion planning, real-time, test vehicle, trajectory planning, urban driving.

I. INTRODUCTION

AUTONOMOUS vehicles will substantially change the personal mobility of the future. This is inter alia because of the reduction of traffic accidents, savings in energy, parking space as well as people's time and the increased access to mobility [1]. But until this vision comes reality, first more research and development work needs to be done in order to solve the most difficult traffic scenarios. These challenging situations occur especially in urban areas because of limited available space, area-specific traffic rules and high density of other traffic participants.

To act autonomously, a vehicle needs three main capabilities: perception, planning and control [2]. In perception, the vehicle collects information about its environment and localizes itself. In planning, the future movement of the vehicle (i.e., trajectory) is determined to achieve a higher order goal such as reaching a specified target region without colliding with other obstacles. Finally, in the control task the planned action is executed. Therefore, planning can be seen as the core competency of the autonomous vehicle. Here, all previously calculated information are combined to one future plan of action. This plan of action needs to be executable by the controller, comfortable, traffic rule compliant and risk-minimizing. Furthermore, it is essential for the acceptance of autonomous vehicles since its result is directly perceived by its passengers. Here, "it should be borne in mind that people are more likely to forgive mistakes made by other people than mistakes made by machines" [1, p. 6].

Solving the complex planning problem in real-time is challenging. To tackle this complex problem, a common planning architecture is chosen as presented in [3] and shown in Fig. 1. In the first layer, the route planning, waypoints through a street network are planned, similar to a navigation systems in current vehicles. In the second layer, the behavior planning, a high-level behavior (also: maneuver) for the autonomous vehicle is planned. Such a maneuver

The review of this article was arranged by Associate Editor Abel C. H. Chen.

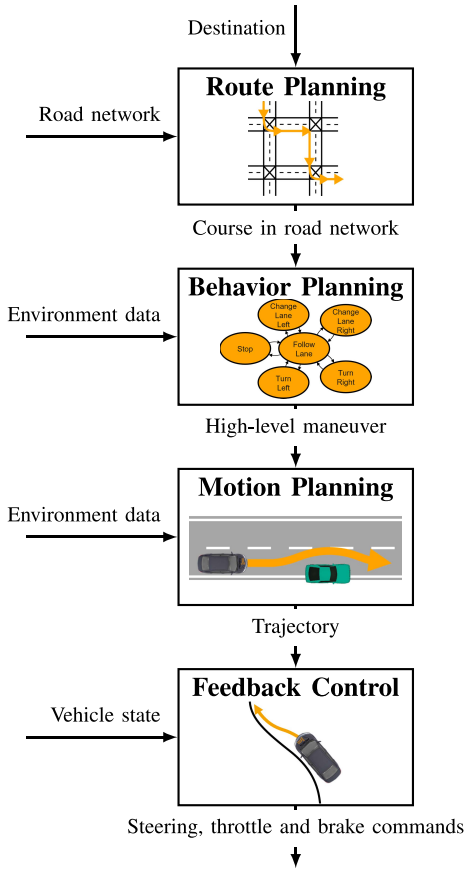


FIGURE 1. Common planning architecture of autonomous vehicles. Figure adapted from [3].

can be e. g. to follow/change the lane or to turn left/right at intersections. In the third layer, the motion planning, the high-level behavior will be transformed into a precise plan of motion, the calculated trajectory. Finally, in the fourth layer, the feedback control, the calculated trajectory will be executed using a local feedback controller.

Therefore, trajectory planning can be seen as the combination of maneuver and motion planning. This separation brings the benefit of reducing the planning complexity, since the motion planning can focus on a specific maneuver. But it may lead to synchronization problems between those two layers, it needs to be ensured that the trajectory reflects the planned maneuver. Additionally, the question arises how the discrete set of maneuvers needs to look like in order to cover all possible scenarios. Especially, in urban driving with narrow roads, limited space and possibly endless maneuvers, it is hard to find a complete set of maneuvers. Therefore, solutions exists solving the planning task of both layers in a single step [4], [5]. Because of the increased complexity, efficient algorithms are needed for real-time execution.

Current planning algorithms all have different strengths and weaknesses, so the search continues for new possible solutions that may surpass existing algorithms. One promising approach is the Monte-Carlo tree search (MCTS) [6]. It is used for solving discrete decision problems, especially in complex combinatorial games. Here, the game Go is considered the most challenging because of its high branching factor. By combining MCTS with reinforcement learning, AlphaGo [7] succeeded in defeating a Go world champion, a milestone for artificial intelligence.

MCTS has been used for behavior planning [8], [9], [10] and end-to-end learning [11]. The idea of this paper is to apply the algorithm to the problem of trajectory planning. Therefore, in this paper we present the CarPre trajectory planner (Monte-**Car**lo Model **Pre**dictive Trajectory Planner) which is based on MCTS, uses a kinematic motion model, and is real-time capable. Since this is an anytime algorithm, it can be guaranteed that a solution is available within the given calculation time. This is especially important for safety-critical applications such as trajectory planning. With possible options, such as e. g. the incorporation of explicit behavioral rules [12], the incorporation of discrete variables into the decision process as well as a variety of possible adaptations, MCTS offers advantages for trajectory planning. Before presenting the novel CarPre trajectory planner, a short overview over the related work is given.

A. RELATED WORK

For the motion planning, three major solution classes exist [3]: optimization, graph-search, and incremental search methods. In the first class, the planning problem is formulated as continuous optimization as in model predictive control (MPC). Using a motion model, the vehicle kinematics can be taken into account as optimization constraints. Common examples are [13], [14], [15], [16]. The advantage of these methods is the (mostly) fast solution in the continuous planning space. Disadvantages are that a solution is only available when the optimization has converged. This can be problematic, especially in complex, critical situations, when an individual optimization requires significantly more computing time than expected (e. g. peaks in Fig. 12 from [15]). Another disadvantage is the convergence to a local minimum. Thus, a higher planning level, such as behavior planning, is mandatory for the optimization to converge to the desired local minimum. Moreover, discrete states (e. g. a traffic light that decides between *stop* and *drive*) cannot be directly incorporated.

For the second class, a graph is generated by discretizing the planning space into a finite number of states. Then, graph search algorithms, like Dijkstra [17] or A* [18], can calculate the optimal trajectory in the discretized graph. To generate such a graph different methods exist. The most common variant is to sample the planning space, resulting in a tree structure [19], [20] or a lattice structure [5], [21]. The advantage of graph-based approaches is that the discretization allows global searching of the graph. By performing a global search, the planning does not get stuck in local minima, but the (discretized) global minimum is found. Moreover, no explicit enumeration of planning constraints is needed as for the continuous optimization methods. Instead, the discretization allows checking of very complex constraints for possible

trajectories or for each trajectory point [22]. The disadvantage of this is the discretization: if it is chosen too large, the smaller planning space simplifies the planning problem, however, no solution may be found, although one exists in the continuous space (e. g. a narrow passage which cannot be passed due to a too coarse discretization). On the other hand, if the discretization is chosen too small, the graph explodes due to the curse of dimension [23].

The third class, the incremental search methods, tries to circumvent the disadvantage of a fixed discretization as in graph-based approaches. This is achieved by incrementally sampling the planning space with finer and finer discretization steps and simultaneously trying to find a solution [3]. Examples include RRT [24], RRT* [25], or CL-RRT [22]. The advantage of these approaches is that a solution is found if it exists (cf. graph-based approaches). However, the computation time required for finding a solution is unbounded, i.e., it is possible that the solution will be found only after far too much time. This usually excludes a real-time application.

B. CONTRIBUTION AND OVERVIEW

Our contribution in this article is two-folded. First, we derive a discretization of possible vehicle movements. Second, MCTS is adapted to the problem of trajectory planning.

The remainder of this paper is organized as follows: First, the used motion model is introduced in Section II which will be the basis for our further examinations. Then, MCTS is introduced in Section III, before the CarPre trajectory planner is presented in Section IV. For this, the state space is first discretized before adapting the MCTS algorithm to the problem of trajectory planning. Finally, the novel trajectory planner is evaluated in Section V in a real test vehicle before concluding the paper in Section VI.

II. MOTION MODEL

A basic requirement for the planning of motion sequences, like a trajectory of car-like vehicles, is a motion model. This motion model describes the relationship between a current state and possible future states. With this, physical movement restrictions can be taken into account in the planning task.

In the literature of trajectory planning, multiple motion models exist. This can be e. g., a simple model of constant turn rate [26], which considers the position and orientation of the vehicle or a single track model [14], [27], [28], which represents the non-holonomic motion of the vehicle. While the kinematic single track model assumes slip-free driving, the dynamic single track model [29], [30] considers the forces between the tires and the road surface. Therefore, it is especially useful for driving at the vehicle's limit.

To decide which motion model should be used, the specific use cases have to be considered. The trajectory planning of a vehicle, which is supposed to drift or to drive best times on a race track, requires a different motion model than e. g. an autonomous cab service. Since the focus of this work is on trajectory planning in urban scenarios, a dynamic vehicle model can be omitted. Instead, ease of use is needed.

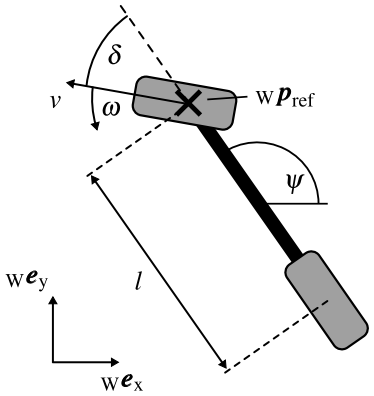


FIGURE 2. Definition of states and parameters in the single track model.

Since the kinematic single track model is the most simple model which considers the non-holonomic movement of a car-like vehicle, it is chosen. In addition, Polack et al. [31] proved that the kinematic single track model can be used for consistent planning if limiting the lateral acceleration to $0.54\mu g$.

As model inputs, the acceleration a as well as the steering rate ω is chosen. With this, the state vector $\boldsymbol{x} = [\,{}_Wx_{ref},\ {}_Wy_{ref},\ \psi,\ v,\ \delta\,]^\top$, consisting of the 2D position ${}_W\boldsymbol{p}_{ref}$ in a world coordinate frame, the orientation ψ, the speed v and the steering angle δ, can be calculated using the following ODE:

$$ {}_W\dot{x}_{ref} = v\cos(\psi+\delta), \tag{1} $$

$$ {}_W\dot{y}_{ref} = v\sin(\psi+\delta), \tag{2} $$

$$ \dot{\psi} = \frac{v\sin\delta}{l}, \tag{3} $$

$$ \dot{v} = a, \tag{4} $$

$$ \dot{\delta} = \omega. \tag{5} $$

With this model, δ can change for a constant input $\boldsymbol{u} = [\,a,\ \omega\,]^\top$ which leads to trajectories with variable curvature. The reference point of the motion model, i.e., the origin of the vehicle-fixed coordinate frame, is chosen on the front axle as discussed in [32]. Because of this, the only parameter of the model is the wheelbase l, which defines the length between the rear and the front axle. All states, inputs and parameters are shown in Fig. 2.

III. MONTE CARLO TREE SEARCH

In this section, a short introduction to Monte Carlo Tree Search (MCTS) is given. MCTS [6] is a search algorithm to solve discrete decision problems. In discrete decision problems, one can choose between a finite number of actions. Examples include board games such as tic-tac-toe, Go, or chess. To solve such a problem, the algorithm builds a search tree, which consists of nodes (states of the search space) and edges (possible actions in a state). For each state, the number of previous visits n_v is stored, as well as the sum of its reward estimates R_s. By trying out different actions, the most promising one is searched, i.e., for the action with the

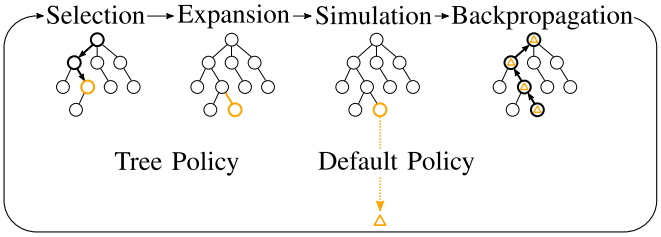


FIGURE 3. Overview of the four steps of one MCTS iteration [33].

highest mean reward estimate $R_\mathrm{m} = R_\mathrm{s}/n_\mathrm{v}$. An iteration of the algorithm consists of four steps, which are shown in Fig. 3:

I *Selection:* In this phase, a child node is selected to be explored in this iteration. It is strategically chosen between actions which are so far most promising (exploitation) and between less promising but also not often investigated actions (exploration). The latter should also be investigated, as they may have given poor results so far due to random evaluation, although they are actually promising. To solve the exploitation/exploration problem, the UCT formula (upper confidence bound applied to trees) is often used:

$$\mathrm{UCT} = \frac{R_{\mathrm{s},j}}{n_{\mathrm{v},j}} + c \cdot \sqrt{\frac{\ln n_\mathrm{v}}{n_{\mathrm{v},j}}}, \tag{6}$$

where n_v is the number of visits to the current node, $n_{\mathrm{v},j}$ and $R_{\mathrm{s},j}$ are the visits and the summed reward estimate of the child node by selecting the jth action, respectively. Changing the parameter $c > 0$, the focus of the selection can be put more on exploitation or more on exploration. Starting from the root node of the tree, the subsequent state to the action with the highest UCT value is selected. This scheme is iteratively executed through all levels of the tree until an action is selected for which no child node yet exists. This approach is called tree policy.

II *Expansion:* The matching child node is created for the selected action from step I.

III *Simulation:* The reward of the selected action is estimated by a simulation. According to a defined default policy (e. g. a random selection of possible actions), actions are selected and simulated until a state is reached where no further actions are available (end of game). This end state is evaluated and passed to the next step.

IV *Backpropagation:* The estimated reward R of the simulation is fed back through the tree starting at the selected node and ending at the root node. To do this, at each of these nodes, the number of visits n_v and the sum of the reward estimates are updated as follows:

$$n'_\mathrm{v} = n_\mathrm{v} + 1, \tag{7}$$
$$R'_\mathrm{s} = R_\mathrm{s} + R. \tag{8}$$

As soon as a defined time t_calc has elapsed or a certain number of iterations has been executed, the algorithm ends. Then, the most promising action of the root node is chosen. For a summary of the properties as well as possible modifications, please refer to [33].

In the following, the modifications of the CarPre-trajectory planning are presented.

IV. CARPRE TRAJECTORY PLANNER

In this section, our novel planning approach, the CarPre trajectory planning, is presented. Based on MCTS, a trajectory is calculated using a discretized search tree. Each tree node represents a vehicle state $\boldsymbol{x}$ in a given environment. Using the kinematic single track model of Section II, a state $\boldsymbol{x}_k$ can be transformed into a new state $\boldsymbol{x}_{k+1}$ by applying a chosen input $\boldsymbol{u}_k$ for a sampling time T_in. Therefore, our approach can be categorized as a graph search method (cf. Section I-A) which creates an equitemporal graph, i.e., a graph with the same time difference between two adjacent tree levels. In the following, the MCTS adaptions for the CarPre planner are presented.

A. DISCRETIZATION OF THE PLANNING SPACE

The trajectory planning problem is in general a continuous planning problem. However, MCTS solves discrete decision problems. Therefore, the action as well as the state space is discretized in the following. To ensure that only physically possible trajectories will be planned, the vehicle motion is constrained by the kinematic single track model (cf. (1)–(5)), so that an action consists of an acceleration value as well as a steering rate. Building an equitemporal search tree, the following discretization depends on the chosen sampling time T_in. In the following, it is assumed that the vehicle can only move forward ($v \geq 0$).

First, the longitudinal states (i.e., acceleration and speed) are discretized equidistantly. The discrete set of acceleration values is calculated according to the discretization distance Δa as well as the minimum a_min and the maximum acceleration a_max as

$$a_\mathrm{d} \in \{a_\mathrm{min} \leq p \cdot \Delta a \leq a_\mathrm{max} \,|\, p \in \mathbb{Z}\}. \tag{9}$$

Because of the fixed sampling time T_in as well as a_d as multiples of Δa, the set of equidistant speeds results in

$$v_\mathrm{d} \in \{p \cdot \Delta a\, T_\mathrm{in} \leq v_\mathrm{max} \,|\, p \in \mathbb{N}_0\}, \tag{10}$$

until a maximum speed v_max is reached. By an appropriate choice of Δa, common behavioral patterns such as emergency braking, comfort braking, coasting down, constant speed, comfort acceleration, and emergency acceleration [34, p. 116] can be represented.

For the lateral motion states (i.e., steering angle and steering rate), the speed-dependent steering angle transformation from [35] is used:

$$|\delta_\mathrm{max}(v)| = \min\left(\arcsin(\kappa_\mathrm{max}\, l),\, \arcsin\left(\frac{a_\mathrm{lat,max}\, l}{v^2}\right)\right), \tag{11}$$

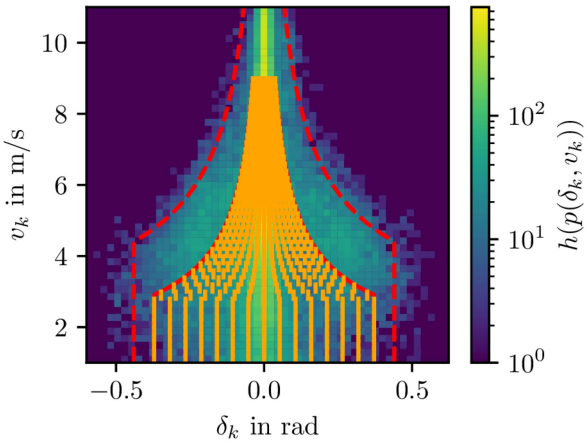


FIGURE 4. Discretization of the steering angle δ compared to the extracted human driving data from [35]. The human driving data is displayed as 2D histogram of δ_k in rad plotted against v_k in m/s with a logarithmic scale. The boundaries extracted in [35] with $\kappa_{\max} = 0.15\text{m}^{-1}$ and $a_{\text{lat,max}} = 2.96\text{m/s}^2$ (dotted red lines) are adapted for the CarPre discretization (red lines) as the overall system should be more comfortable. The resulting equidistant discrete steering angles $\delta_d(v_d)$ are shown in orange for $n_\delta = 15$, $l = 2.79\,\text{m}$, $\kappa_{\max} = 0.13\text{m}^{-1}$ and $a_{\text{lat,max}} = 1.3\text{m/s}^2$.

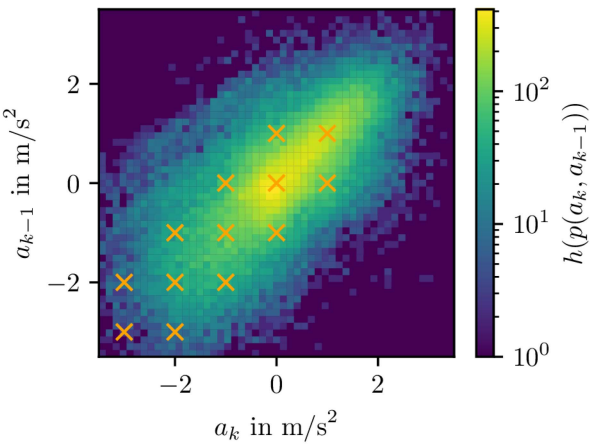


FIGURE 5. Discretization of the acceleration a compared to the extracted human driving data from [35]. The human driving data is displayed as 2D histogram of the acceleration a_k in m/s^2 plotted against the previous chosen acceleration a_{k-1} in m/s^2 with a logarithmic scale. The equidistant discrete accelerations of (9) are combined with the jerk limitation of (16) and are shown as orange crosses for $\Delta a = 1\,\text{m/s}^2$, $a_{\min} = -3\,\text{m/s}^2$ and $a_{\max} = 1\,\text{m/s}^2$.

where $\kappa_{\max}$ is the maximum drivable path curvature, $a_{\text{lat,max}}$ is the maximum lateral acceleration, and l is the wheelbase of the vehicle. This transformation was derived from human driven trajectories, limits the steering angle with the help of its two variable parameters and is shown in Fig. 4 as red dotted lines. For low driving speeds, the maximum drivable path curvature $\kappa_{\max}$ of the vehicle limits the steering angle. For higher speeds, the maximum lateral acceleration $a_{\text{lat,max}}$ takes over the limitation. This ensures that lateral accelerations do not become too large and planned trajectories can be executed by the feedback controller at all times. The speed-dependent steering angle $\delta_d(v_d)$ with $n_\delta \in \{2p-1 \,|\, p \in \mathbb{N}\}$ discretized states is set to

$$\delta_d(v_d) \in \left\{ q \cdot \Delta\delta(v_d) \,\middle|\, q \in \mathbb{Z} \;\wedge\; |q| < \frac{n_\delta}{2} \right\}, \tag{12}$$

$$\text{with } \Delta\delta(v_d) = \frac{2 \cdot |\delta_{\max}(v_d)|}{n_\delta - 1} \tag{13}$$

and $|\delta_{\max}(v_d)|$ taken from (11). An example discretization is shown in Fig. 4. The discrete steering rate $\omega_d(\delta_{d,i}(v_d), a_d)$ with $n_\omega \in \{2p-1 \,|\, p \in \mathbb{N}\}$ discretized states is derived from the current steering angle $\delta_{d,i}(v_d)$:

$$\omega_d(\delta_{d,i}(v_d), a_d) \in \left\{ q \cdot \frac{\delta_{d,i+q}(v_d + a_d T_{\text{in}}) - \delta_{d,i}(v_d)}{T_{\text{in}}} \,\middle|\, q \in \mathbb{Z} \;\wedge\; |q| < \frac{n_\omega}{2} \;\wedge\; 0 \le i+q < n_\delta \right\}. \tag{14}$$

With this, a state space symmetrical around zero is obtained for the lateral motion. Furthermore, the discretization leads to a state lattice, i.e., solving the equations of motion (4) and (5) with the discretized inputs a_d, $\omega_d(\delta_{d,i}(v_d), a_d)$ as well as the sampling time T_{in} leads to the states v_d, $\delta_d(v_d)$ without discretization errors. For the remaining states of the kinematic vehicle model (position, orientation), this is not possible due to the nonlinear functions in (1) to (3). Therefore, a discretization of these states can only be performed with a discretization error. This error can only be avoided if the respective states remain in the continuous range of values. In general, this is possible because MCTS needs only discrete actions and no discrete states.

Using the discretization described above, a search tree can be constructed as described in Section III. When selecting the discretization parameters, it should be noted that they determine the complexity of the planning space. Especially the number of inputs $n_{\text{in}} = n_a n_\omega$ as well as the sampling time T_{in} are crucial, since they determine the number of possible action combinations $n_{\text{in,total}}$ within a planning horizon $T_{\text{hor}} \in \{p \cdot T_{\text{in}} \,|\, p \in \mathbb{N}\}$. The number of combinations increase exponentially (cf. curse of dimension [23]):

$$n_{\text{in,total}} = (n_a n_\omega)^{\frac{T_{\text{hor}}}{T_{\text{in}}}}. \tag{15}$$

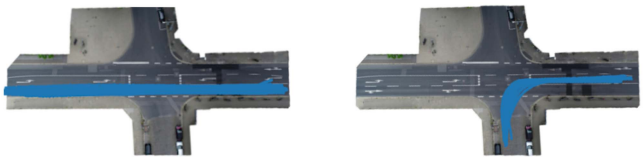
To mitigate this exponential growth, the number of possible actions can be limited depending on the last selected action. An example for this is the limitation of the acceleration a_k depending on the last acceleration a_{k-1}:

$$a_k \in \{a_{k-1} - \Delta a;\; a_{k-1};\; a_{k-1} + \Delta a\}, \tag{16}$$

so that the number of possible acceleration actions n_a is reduced to three. This restriction resembles a limitation of the jerk, thus increasing the driving comfort and reducing the planning complexity at the same time. A resulting example discretization of the acceleration is presented in Fig. 5.

B. END OF GAME

The end of game specifies states at which no further action can be selected. This applies to states in the decision tree as well as states in the simulation for the reward estimation. In board games, such an end of game is determined by the rules of the game. In the case of trajectory planning, reaching a goal state can be the end of game. However, this goal state is usually farther away and therefore only tracked by route



(a) Paths for driving straight. (b) Paths for turning.

FIGURE 6. Selected paths (blue) of human drivers. Drivers keep the lane that takes them to their destination according to route planning. Thus, the default behavior of drivers is mostly to keep the lane. The paths as well as the background image are taken from the inD dataset [37].

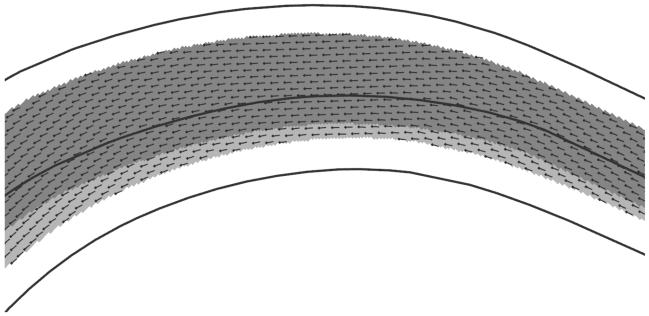


FIGURE 7. Discretized direction map (black direction pointers) for selecting steering rates for lane keeping. Such a vector field can be created using a road map or a sensor-based environment model (black lines). The lane regions are divided into the own lane (light gray) as well as the opposite lane (dark gray) and are reduced by the space required by the ego vehicle. The required space is calculated as in the point model of [32].

planning (cf. Fig. 1). Thus, additional end states must be defined for the trajectory planner:

1) Reaching a defined time horizon $t = T_{\text{hor}}$ (also: planning horizon).
2) Leaving the area for which environmental information is available, so that no further planning is possible.
3) Causing a collision with other objects.
4) Stopping of the vehicle for $t > 0$.

The limitation of the planning horizon 1) is similar to the principle of the receding horizon control [36], which is used in MPC. Thus, planning considers only a limited time window to determine the next trajectory. After a trajectory is computed, the first part of it is executed before a new schedule is started with a shifted time window.

C. DEFAULT POLICY

In order to evaluate a chosen action, a reward value is estimated by means of simulation. To increase the algorithm's time of convergence, random combinations of acceleration and steering rate values for the simulation are avoided. Instead, a default policy is presented below, which is derived from human behavior. This behavior is based on an analysis of the inD dataset [37]. In general, two main behavior patterns can be identified in the dataset. First, human drivers try to follow their lane (cf. Fig. 6). Second, they usually accelerate only for a longer period of time (cf. Fig. 5). Therefore, the default policy is defined as follows:

- *Acceleration:* The previous acceleration value is chosen again for the next k_{acc} time steps (constant acceleration model). Subsequently, acceleration values around zero (i.e., $a_{\text{d}} \in \{-\Delta a;\ \Delta a\}$) are set to $a_{\text{d}} = 0$.
- *Steering rate:* The steering rate is set to zero for the next k_{ω} actions. Subsequently, the steering rate is selected based on a direction map so that the vehicle follows the course of the road as best as possible (lane keeping). Such a direction map is shown in Fig. 7.

D. HEURISTIC FOR INITIAL NODE REWARD

According to the UCT formula (6), any action that has never been selected is preferred over an already explored action because of

$$\lim_{n_{\text{v},j}\to 0} \text{UCT} = \lim_{n_{\text{v},j}\to 0} \left(\frac{R_{\text{s},j}}{n_{\text{v},j}} + c \cdot \sqrt{\frac{\ln n_{\text{v}}}{n_{\text{v},j}}} \right) = \infty. \quad (17)$$

This means that all possible actions of a node are explored first before a node of the next tree level can be explored. With limited computational time, this can lead to the search tree not reaching the required depth for planning a good trajectory. To speed up the convergence of the algorithm, an initialization of the node reward values $R_{\text{s},j} = R_{\text{ini}}$ is done. Similarly, the initial number of visits of the node is set to $n_{\text{v},j} = 1$. Using the direction map, the best steering rate ω_{best} for lane keeping is determined for the current state, and accordingly the initialization R_{ini} is set to

$$R_{\text{ini}} = \begin{cases} R_{\text{ini},1} & \text{for } \omega_k = \omega_{\text{best}} \text{ and } a_k = a_{k-1}; \\ R_{\text{ini},2} & \text{for } \omega_k \neq \omega_{\text{best}} \text{ and } a_k = a_{k-1}; \\ 0 & \text{else,} \end{cases} \quad (18)$$

where $0 \leq R_{\text{ini},2} < R_{\text{ini},1} \leq 1$. With this choice, an initial preference of future actions is set, so that actions with equal accelerations as well as steering rates to follow the road are preferred during exploration. When selecting a particular action, the default policy is used to simulate to the end of game and the reward estimate is backpropagated. By averaging, the initial value loses influence with increasing iterations and converges to the real reward value.

E. REWARD FUNCTION

If an end of game is reached, the planned trajectory must be evaluated to get an estimation of the chosen action. In order to establish a clear prioritization of the planning goals, a reward function $R(\boldsymbol{x}_k)$, which differs for each reward term by one order of magnitude, is chosen for each state $\boldsymbol{x}_k$ of the trajectory:

$$R(\boldsymbol{x}_k) = \frac{1}{\sum_{i=0}^{m-1} b_{\text{R}}^i} \sum_{i=0}^{m-1} b_{\text{R}}^i R_i(\boldsymbol{x}_k) \quad (19)$$

with the m reward terms $R_i(\boldsymbol{x}_k) \in [0, 1]$ and the basis of the order of magnitude b_{R}. The overall reward value of one trajectory R_{traj} consists of the normalized sum of all $n_{\text{traj}} = T_{\text{hor}}/T_{\text{in}}$ possible states:

$$R_{\text{traj}} = \frac{1}{n_{\text{traj}}} \left(\sum_{k=1}^{p} R(\boldsymbol{x}_k) + \left(n_{\text{traj}} - p\right) R(\boldsymbol{x}_p) \right), \quad (20)$$

where $p \leq n_{\text{traj}}$ describes the number of trajectory points to be evaluated. If the simulation is terminated early ($p < n_{\text{traj}}$) by the end of game described in Section IV-B, the state evaluation of the final state $\boldsymbol{x}_p$ is repeated until n_{traj} points have been incorporated into the trajectory evaluation. This is important so that each state $\boldsymbol{x}_k$ is weighted equally in each trajectory evaluation.

An exemplary reward term is the term for reaching a target speed v_{target}. It can be chosen to

$$R_{\text{v}}(\boldsymbol{x}_k) = \begin{cases} 1 - \frac{|v|}{v_{\max}} & \text{if collision or outside} \\ & \text{of drivable area;} \\ 1 - \frac{|v_{\text{target}} - v|}{v_{\max}} & \text{else.} \end{cases} \quad (21)$$

As seen in this example, the trajectory discretization allows checking of complex constraints dependent on discrete events. In this case, the target speed is changed to zero if the vehicle collides or leaves the allowed drivable area, since this is the desired behavior. These properties, i.e., the evaluation of complex reward terms with discrete decision variables combined with the weighting of different order of magnitudes, lead to a flexible specification of the desired behavior. Therefore, a behavior-semantic scenery description [12] can be integrated in the CarPre trajectory planner to implement the behavioral rules of the road.

F. EXTRACTION OF THE PLANNED TRAJECTORY

After the defined calculation time t_{calc} has elapsed, the newly planned trajectory Π_k can be extracted from the created search tree. To avoid small variations between two successive trajectories, the actions of the last planned trajectory Π_{k-1} are used, if available. Starting at the root node of the recomputed search tree, the j_{best} action is selected in each node, which has the largest mean reward $R_{\text{m,best}}$. If the trajectory extracted so far is still equal to the previous one, the current reward value $R_{\text{m,best,prev}}$ of the previous best $j_{\text{best,prev}}$ action is also evaluated. If these two reward values are within a tolerable deviation ϵ_{R}, the previous best action is preferred over the current best action. This results in

$$j_{\text{best}} = \begin{cases} j_{\text{best,prev}} & \text{for } |R_{\text{m,best}} - R_{\text{m,best,prev}}| < \epsilon_{\text{R}} \text{ and so} \\ & \text{far extracted } \Pi_k \text{ equals } \Pi_{k-1}; \\ j_{\text{best}} & \text{else.} \end{cases} \quad (22)$$

This procedure is executed, starting at the root node, until either n_{traj} trajectory points are extracted or a leaf node, i.e., a node without further child nodes, is reached.

V. EVALUATION

In order to evaluate the presented planning approach, a lane following maneuver is chosen. Here, the planner's goal is to follow its own lane with (if possible) the target speed. For this, the algorithm is implemented in C++ and executed on an Intel Xeon E5-2698 v4 processor. To speed-up the exploration progress, the algorithm is parallelized on 10 threads using tree parallelization with local mutexes (cf. [38]). The planning algorithm is then put into operation on a real test vehicle which is shown in Fig. 8 using the minimal software architecture presented in Fig. 9. In this architecture, planning is only based on a road model calculated from an HD map. The tests are executed on the available test-track, the August-Euler airfield (cf. Fig. 10) with the used parameters listed in Table 1. The reward function of the algorithm includes terms for avoiding collisions, reaching the target speed, following the own lane and minimizing accelerations. Because of safety reasons, the vehicle is only allowed to drive

TABLE 1. CarPre parameters.

Parameter	Description	Value
$a_{\text{lat,max}}$	max. lateral acceleration	1.3 m/s^2
$a_{\max}$	max. acceleration	1 m/s^2
$a_{\min}$	min. acceleration	-3 m/s^2
Δa	acceleration difference	1 m/s^2
c	UCT value	0.5
$\kappa_{\max}$	max. curvature	0.13 m^{-1}
n_δ	number discretized $\delta_{\text{d}}(v_{\text{d}})$	15
n_ω	number discretized $\omega_{\text{d}}(\delta_{\text{d},i}(v_{\text{d}}), a_{\text{d}})$	3
t_{calc}	MCTS calculation time	0.1 s
T_{hor}	planning horizon	6 s
T_{in}	sampling time	0.2 s
v_{target}	target speed	8.4 m/s

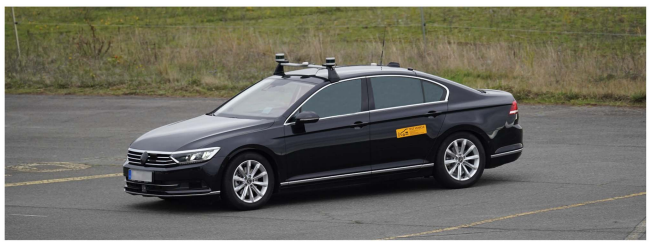


FIGURE 8. PRORETA 5 test vehicle: Volkswagen Passat (B8).

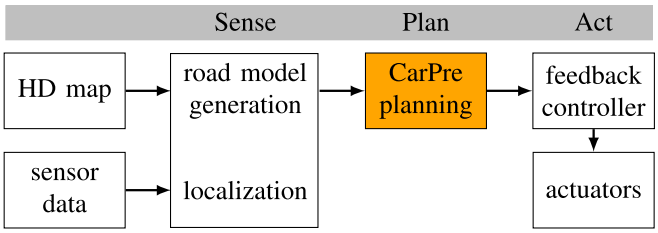
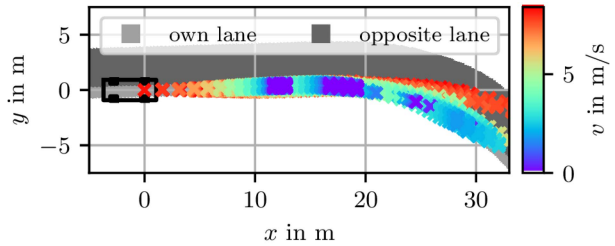
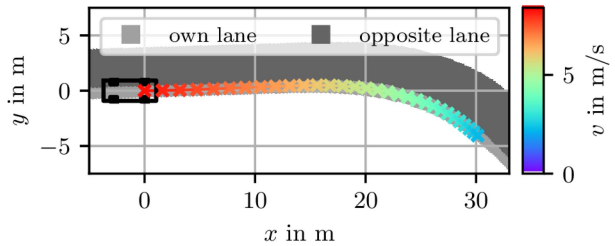


FIGURE 9. Minimal software architecture for planner evaluation.

FIGURE 10. Available test track (orange) at the August-Euler airfield of the TU Darmstadt (image taken from Google Earth Pro). The curves of the track are numbered for evaluation.



(a) Monte Carlo search tree with 12969 nodes.



(b) Best trajectory resulting from the search tree shown in (a).

FIGURE 11. Exemplary CarPre trajectory planning of curve 1. The starting speed of the vehicle is too high for the depicted curve (cf. red/orange nodes with high speeds in the right area of (a)), so that the vehicle has to brake to avoid missing the turn. Due to the kinematic single track model as well as the speed-dependent steering angle discretization, only comfortable, controllable motions can be explored. Thus, in this example, different braking maneuvers are explored before the trajectory shown in (b) is extracted from the search tree.

8.4 m/s = 30.24 km/h. Furthermore, a steering torque limitation exists, so that a safety driver can overrule the planned trajectory at all times.

First, a single planning step of a right turn is analyzed (Fig. 11). Here, the vehicle has already reached the target speed v_{target} and is approaching the curve. However, the curve is too tight so that the vehicle has to brake. Due to the speed-dependent steering angle discretization, the vehicle can only plan within the given physical constraints, i.e., if the vehicle does not brake, it would miss the turn (cf. red/orange dots in the right of Fig. 11a). Without reducing v_{target}, the algorithm searches for the fastest possible speed to safely drive through the curve. Due to the initialization of the possible actions (cf. Section IV-D) as well as the limited computation time, the vehicle slows down more than necessary (cf. light blue dots in the right of Fig. 11b). This is because of the exploration characteristic of the algorithm, the actions in the near future (i.e., directly in front of the vehicle) are well estimated by the high number of simulations. As a result of the branching of the tree, the behavior in the far future is largely determined by the initialization of the action values and the default strategy. However, using the receding horizon control, only the first planned action is executed before the planning for the next time step begins.

In order to analyze the planning over multiple planning steps and the driving comfort, the test vehicle drives on the test track for ~45 minutes (one planning cycle every 0.2 s, in total 13513 cycles). The goal is to follow the route shown in Fig. 10 with the given target speed of 8.4 m/s = 30.24 km/h. During the measurements, the speed-dependent steering torque limit of the vehicle is exceeded several times, although the dynamics of the vehicle were not at the limit (five times in curve 5, and once in curve 2). In this case, the automation stops, the vehicle is brought to a standstill by the safety driver and the automation is enabled again. Such an overrun is not desirable, but a later fully automated system will not include such a limit. Fig. 12 shows an almost complete lap on the test track, only the long, straight part between curves 1 and 5 (lower straight line in Fig. 10) is not shown. In this example, the steering torque limit is exceeded in curve 5, which is marked in red.

In general, the feedback controller, in combination with the inertia of the actuators, smoothes the planned trajectory consisting of the discrete-time inputs shown in blue. Individual small outliers in the longitudinal acceleration a of the scheduled trajectory such as between 30 and 40 s or 130 and 140 s are filtered out. Although such high-frequency fluctuations are undesirable and should be avoided by further adjustments, they are not noticed by the driver.

The lateral acceleration a_{lat} of the vehicle oscillates slightly around zero when driving straight ahead, but this is not perceived in the vehicle. In the curves, the steering angle discretization limits the maximum lateral acceleration, which is maintained by the vehicle. Additionally, the lateral deviation d_{lat} of the vehicle to a virtual center line is shown. Since d_{lat} is not part of the cost function, it will not be optimized. The deviations in curve 2 and 5 are high, because these curves are on intersections with wide lane segments. Maximizing the speed with limited lateral accelerations results in driving along the inner edge of the curve and thus, high deviations from the center line. The observed behavior is similar to driving a racing line without leaving the own lane. Furthermore, the vehicle already starts to brake before a curve, comparable to human driving behavior. The required speed is then reached within the curve before accelerating again towards the end of the curve.

Fig. 13 shows the G-G diagram of the planned and driven trajectory. Due to the physically constrained discretization, only actions within the selected boundaries can be chosen. Therefore, the circle, representing $1/3\,G$, is not reached. All in all, this creates a natural driving experience.

VI. CONCLUSION

In this article we present CarPre trajectory planning, a novel planning algorithm based on MCTS. Using the kinematic single track model in combination with a speed-dependent steering angle transformation, possible movements of the vehicle are described and limited by physical constraints. With this, a discrete action space, consisting of discrete-value pairs of accelerations and steering rates, is derived which contains only drivable motions. This action space is then used to create an equitemporal search tree. Monte-Carlo simulations in combination with heuristics estimate the long-term effect of each short-term action. Thus, high-level maneuvers can be indirectly incorporated in the algorithm and do not need to be calculated beforehand by a behavior planning module. Furthermore, the algorithm offers great

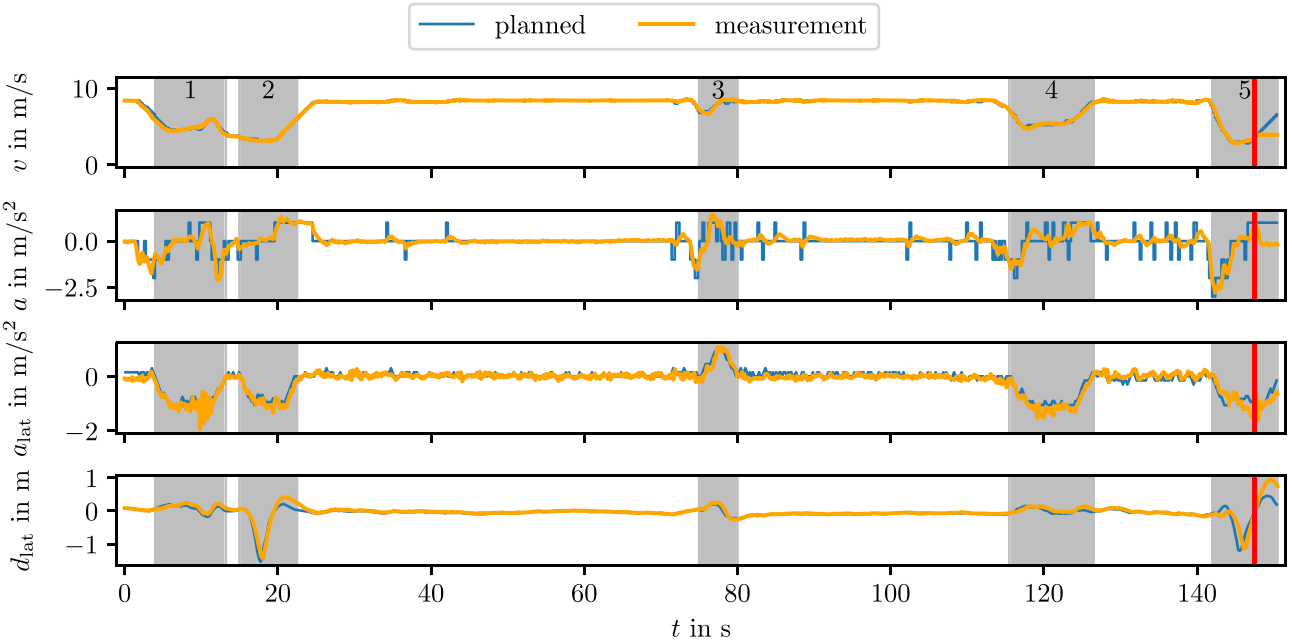


FIGURE 12. Planned trajectory (blue) vs. measured values of the driven trajectory (yellow). The curves marked in Fig. 10 are highlighted in gray and numbered accordingly. Exceeding the steering torque limit in curve 5 is marked in red. The acceleration measurement values recorded at 50 Hz are smoothed by a low-pass filter (binomial filter of order 40, cutoff frequency: 120 Hz).

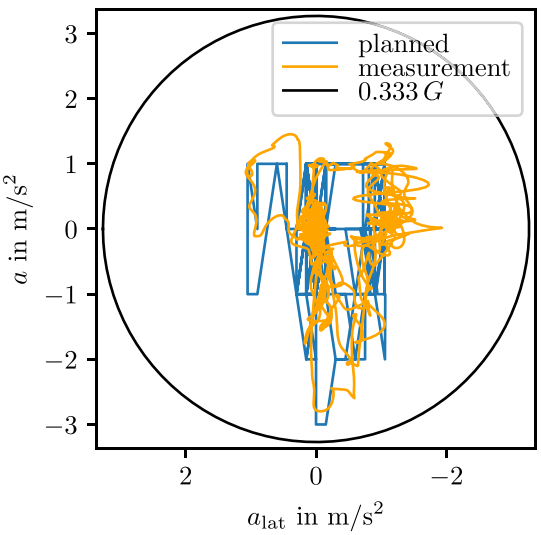


FIGURE 13. G-G diagram of the planned trajectory (blue) vs. the measured values of the driven trajectory (yellow) of Fig. 12. Note the inverted *x*-axis, since positive lateral accelerations represent left turns.

flexibility. Being able to limit and evaluate trajectories at discrete time steps, complex cost functions or certain behavioral constraints can be easily implemented. The practical feasibility of the novel concept is demonstrated by running the algorithm on a real test vehicle. Here, comfortable, foresighted driving suggests future acceptance of the real-time capable algorithm.

ACKNOWLEDGMENT

The authors kindly thank Continental AG for their great cooperation within PRORETA 5, a joint research project of TU Darmstadt, University of Bremen, TU Iaşi and Continental AG to investigate future concepts for intelligent and learning driver assistance systems.

REFERENCES

[1] A. Herrmann, W. Brenner, and R. Stadler (Emerald Publ. Ltd., Bingley, U.K.). *Autonomous Driving*. (Mar. 2018). doi: 10.1108/9781787148338.

[2] S. D. Pendleton et al., "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, Feb. 2017, doi: 10.3390/machines5010006.

[3] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: 10.1109/tiv.2016.2578706.

[4] S. Heinrich, "Planning universal on-road driving strategies for automated vehicles," Ph.D. dissertation, Dept. Math. Comput. Sci., Freie Universität Berlin, Berlin, Germany, 2018, doi: 10.1007/978-3-658-21954-3.

[5] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4889–4895, doi: 10.1109/icra.2011.5980223.

[6] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Computers and Games*. Berlin, Germany: Springer, 2007, pp. 72–83, doi: 10.1007/978-3-540-75538-8_7.

[7] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.

[8] D. Lenz, T. Kessler, and A. Knoll, "Tactical cooperative planning for autonomous highway driving using Monte-Carlo tree search," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2016, pp. 447–453, doi: 10.1109/ivs.2016.7535424.

[9] K. Kurzer, C. Zhou, and J. M. Zöllner, "Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2018, pp. 529–536, doi: 10.1109/ivs.2018.8500712.

[10] K. Kurzer, M. Fechner, and J. M. Zöllner, "Accelerating cooperative planning for automated vehicles with learned heuristics and Monte Carlo tree search," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Oct. 2020, pp. 1726–1733, doi: 10.1109/iv47402.2020.9304597.
[11] J. Chen, C. Zhang, J. Luo, J. Xie, and Y. Wan, "Driving maneuvers prediction based autonomous driving control by deep Monte Carlo tree search," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7146–7158, Jul. 2020, doi: 10.1109/tvt.2020.2991584.
[12] F. Glatzki, M. Lippert, and H. Winner, "Behavioral attributes for a behavior-semantic scenery description (BSSD) for the development of automated driving functions," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 667–672, doi: 10.1109/itsc48978.2021.9564892.
[13] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, Jun. 2021, doi: 10.1109/tiv.2020.3017342.
[14] F. Seccamonte, J. Kabzan, and E. Frazzoli, "On maximizing lateral clearance of an autonomous vehicle in urban environments," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1819–1825, doi: 10.1109/itsc.2019.8917353.
[15] Q. Wang, B. Ayalew, and T. Weiskircher, "Predictive maneuver planning for an autonomous vehicle in public highway traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1303–1315, Apr. 2019, doi: 10.1109/tits.2018.2848472.
[16] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha—A local, continuous method," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2014, pp. 450–457, doi: 10.1109/ivs.2014.6856581.
[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/bf01386390.
[18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/tssc.1968.300136.
[19] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4523–4530, doi: 10.1109/iros.2018.8593813.
[20] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, Dec. 2011, doi: 10.1177/0278364911423042.
[21] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3149–3154, doi: 10.1109/itsc.2019.8917032.
[22] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009, doi: 10.1109/tcst.2008.2012116.
[23] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1972.
[24] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001, doi: 10.1177/02783640122067453.
[25] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robot. Sci. Syst. VI*, Jun. 2010, pp. 1–8, doi: 10.15607/rss.2010.vi.034.
[26] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 1681–1686, doi: 10.1109/iros.2008.4651075.
[27] S. Dixit et al., "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2310–2323, Jun. 2020, doi: 10.1109/tits.2019.2916354.
[28] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, Jun. 2017, doi: 10.1109/tits.2016.2614705.
[29] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 2994–3008, Sep. 2018, doi: 10.1109/tits.2017.2771351.
[30] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2015, pp. 1094–1099, doi: 10.1109/ivs.2015.7225830.
[31] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2017, pp. 812–818, doi: 10.1109/ivs.2017.7995816.
[32] C. Popp, C. Ziegler, M. Sippel, and H. Winner, "Ideal reference point in planning and control for automated car-like vehicles," *IEEE Trans. Intell. Veh.*, early access, Mar. 7, 2022, doi: 10.1109/tiv.2022.3156370.
[33] C. B. Browne et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012, doi: 10.1109/tciaig.2012.2186810.
[34] F. Damerow, "Situation-based risk evaluation and behavior planning," Ph.D. dissertation, Dept. Electr. Comput. Eng., Technische Universität Darmstadt, Darmstadt, Germany, 2018. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/6790
[35] C. Ziegler, V. Willert, and J. Adamy, "Modeling driving behavior of human drivers for trajectory planning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20889–20898, Nov. 2022, doi: 10.1109/tits.2022.3183204.
[36] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Syst. Mag.*, vol. 31, no. 3, pp. 52–65, Jun. 2011, doi: 10.1109/mcs.2011.940571.
[37] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at German intersections," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Oct. 2020, pp. 1929–1934, doi: 10.1109/iv47402.2020.9304839.
[38] G. Chaslot, M. J.-B. Winands, M. H. M. Winands, and H. J. van den Herik, "Parallel Monte-Carlo tree search," in *Computers and Games*. Berlin, Germany: Springer, 2008, pp. 60–71, doi: 10.1007/978-3-540-87608-3_6.

CHRISTOPH ZIEGLER received the B.S. and M.S. degrees in electrical engineering and information technology from the Technical University of Darmstadt, Germany, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree. During and after his studies, he worked in a startup on printing machines for printed electronics. He has been a Research Associate with the Department of Control Methods and Intelligent Systems, Technical University of Darmstadt since 2019. His research interests include the analysis of human driving data and trajectory planning.

JÜRGEN ADAMY received the Diploma degree in electrical engineering and the Ph.D. degree in control theory from the University of Dortmund, Germany, in 1987 and 1991, respectively. From 1991 to 1995, he worked as a Research Engineer and from 1995 to 1998, as a Research Manager with the Siemens Research Center, Erlangen, Germany. Since 1998, he has been a Professor with the Technical University of Darmstadt, Germany, and the Head of the Department of Control Methods and Intelligent Systems. His main fields of research are control theory, computational intelligence, and autonomous mobile robots.