Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Search-based task and motion planning for hybrid systems: Agile autonomous vehicles

Zlatan Ajanović [a,c,*], Enrico Regolin [b], Barys Shyrokau [c], Hana Ćatić [d], Martin Horn [e], Antonella Ferrara [b]

[a] Virtual Vehicle Research GmbH, Inffeldgasse 21/A, 8010 Graz, Austria
[b] University of Pavia, via Ferrata 5, 27100 Pavia, Italy
[c] Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
[d] University of Sarajevo, Zmaja od Bosne bb, 71000 Sarajevo, Bosnia and Herzegovina
[e] Graz University of Technology, Inffeldgasse 21/B, 8010 Graz, Austria

ARTICLE INFO

Keywords:
Task and motion planning
Hybrid systems
Autonomous vehicles
Search
Motion planning
Behavior planning

ABSTRACT

To achieve optimal robot behavior in dynamic scenarios we need to consider complex dynamics in a predictive manner. In the vehicle dynamics community, it is well know that to achieve time-optimal driving on low friction surface, the vehicle should utilize drifting. Hence, many authors have devised rules to split circuits and employ drifting on some segments. These rules are suboptimal and do not generalize to arbitrary circuit shapes (e.g., S-like curves). So, the question "*When* to go into *which mode* and *how* to drive in it?" remains unanswered. To choose the suitable mode (discrete decision), the algorithm needs information about the feasibility of different modes (continuous motion). This makes it a class of Task and Motion Planning (TAMP) problems, which are known to be hard to solve optimally in real-time. In the AI planning community, search methods are commonly used. However, they cannot be directly applied to TAMP problems due to the continuous component. Here, we present a search-based method that effectively solves this problem and efficiently searches in a highly dimensional state space with nonlinear and unstable dynamics. The space of the possible trajectories is explored by sampling different combinations of motion primitives guided by the search. Our approach allows to use multiple locally approximated models to generate motion primitives (e.g., learned models of drifting) and effectively simplify the problem without losing accuracy. The algorithm performance is evaluated in simulated driving on a mixed-track with segments of different curvatures (right and left). Our code is available at https://git.io/JenvB.

1. Introduction

Similarly to other Artificial Intelligence (AI) applications that can be modeled as Intelligent Agents, Autonomous Vehicle (AV) control is based on a Sensing–Planning–Acting cycle. In this work, we focus on the Planning aspect of the cycle, with the goal to provide feasible Motion Planning (MP) for agile automated driving on a gravel race track. This is not only an exciting problem that attracts a lot of attention in the motorsports, but also a practical benchmark that pushes to the limits our real-time motion planning capabilities.

Existing MP methodologies usually make trade-offs between model complexity and computation time, which is especially challenging in agile automated driving problem. Although simplified vehicle models enable the development of control strategies easier to implement, when the vehicle is driving near the limits of handling (which is the focus of this research), they may fail to represent properly vehicle dynamics. Because of that, many control strategies either generate trajectories that the vehicle cannot physically follow or they settle with conservative driving and do not exploit full vehicle possibilities, which is not desirable in racing scenarios. This claim is supported by a recent survey on behavior and motion planning for autonomous vehicles by Sharma et al. (2021), who claim that future research directions should consider curvature and vehicle orientation as well as tire–road interaction forces. All of these are the major focus of the presented work. Besides the obvious use for agile automated driving on a race track, the presented system has relevance for crash avoidance, such that the full vehicle dynamics could be employed to avoid collision with other vehicles (Perumal et al., 2021).

Due to different dominant effects in vehicle dynamics, we can distinguish different approaches suitable for different road surfaces (e.g., high and low friction coefficients). High friction surfaces (high

* Corresponding author at: Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands.
E-mail addresses: z.ajanovic@tudelft.nl (Z. Ajanović), enrico.regolin@unipv.it (E. Regolin), b.shyrokau@tudelft.nl (B. Shyrokau), hcatic1@etf.unsa.ba (H. Ćatić), martin.horn@tugraz.at (M. Horn), a.ferrara@unipv.it (A. Ferrara).

https://doi.org/10.1016/j.engappai.2023.105893
Received 1 March 2022; Received in revised form 9 December 2022; Accepted 20 January 2023
Available online 8 February 2023

μ) enable better controllability of the vehicle, and the dynamics can be adequately represented with linearized models. On the other hand, on lower friction surfaces the control action can enter the saturated region where the vehicle dynamics change significantly.

For driving on high μ roads, many different methods were presented so far, and some were even tested on real vehicles (Valls et al., 2018; Betz et al., 2019). Predictive planning of future vehicle trajectories can enable real-time control of driving while avoiding static obstacles (Liniger et al., 2015). Recently, Liniger and Lygeros (2019) extended the approach to racing scenarios with multiple agents (although not real-time). Although these approaches use a nonlinear bicycle model with Pacejka's tire model, the road surface has a high μ, which can be observed as the vehicle is not performing drifting or trail-braking maneuvers. Additionally, this MP approach is based on an exhaustive search and works well only for short horizons (due to exponential complexity). Furthermore, it is well suited only for high friction conditions where fast transitions between constant velocity primitives can be achieved. Approaches like this are not well suited for controlling a vehicle in lower friction conditions. Driving in low friction environments requires longer horizons and a more detailed vehicle model, as the control action often enters the saturated region. Additionally, the solution for minimum-time driving on high μ surfaces minimizes the curvature of the driving path, so the optimal path is on the edge of the road, which is not robust for a gravel-like road.

Another line of work considers driving on gravel-like roads (low μ), e.g., driving with high side-slip angles like drifting, trail-braking, etc. to improve the robustness of the trajectory. Most of the current works in this direction consider two specific scenarios: sustained drift or transient drift. One example of a *transient drift* scenario is drift parking, as shown by Kolter et al. (2010), where the vehicle approaches the empty parking slot with some velocity and enters temporarily a drift state to rotate and slide laterally in the parking slot. On the other hand, in a *sustained drift* scenario, the goal is to maintain steady-state drifting like the well-known phenomena "donut drifting", where the vehicle continuously drifts in circles of small radius. Velenis et al. modeled high side-slip angle driving and showed that for certain boundary conditions it can be achieved as a solution to the minimum-time cornering problem (Velenis et al., 2007, 2008). Tavernini et al. (2013) showed that to achieve minimum time cornering with maximum exit velocity in low-friction conditions the vehicle has to go in agile drifting maneuvers. Although these works provide deep insight, due to computational complexity, they cannot achieve online performance. Based on results generated offline, by using the work of Velenis et al. (2008) and You and Tsiotras (2018) proposed a method for learning the primitive trail-braking behavior offline, and use the learned model to enable online generation of trail-brake maneuvers. This approach decomposes trail-braking into three stages: entry corner guiding, steady-state sliding, and straight-line exiting. A similar approach, based on the decomposition of the problem, was presented by Zhang et al. (2018). This approach divides the horizon into three regions, finds a path for each region (using Rapidly-exploring Random Trees (RRT), rule-based sampling, and Proportional Integral Control), and then concatenates them. In these two approaches, decomposition is rather rule-based and not scalable to different circuits. Besides simulation work, an impressive demonstration of the scaled vehicle drifting is shown by Williams et al. (2017), where a Model-Based Reinforcement Learning (MBRL) approach is employed. After extensive trial and error, the RL agent learns the model and, using extensive parallelization, applies planning to find a feasible trajectory and drive on the given track. However, as for the aforementioned approaches, the considered scenario is relatively simple with only a single curve. Additionally, this approach demands a lot of experience to learn the model and a lot of computational resources to find a feasible plan. An even more impressive demonstration of drifting is presented by Goh et al. (2019), where a full-scale DMC DeLorean vehicle is able to drift. However, Goh et al. (2019) present only a controller that requires a predefined path and not a motion planner. They consider only drifting with a few steady-state drifting options (e.g., right and left), for a given reference path, without straight driving, effectively avoiding the combinatorial problem. Therefore, this approach would have limitations to generalizing outside of the specific track. We use this controller with our proposed planner in this work to improve close-loop robustness in drifting mode. The drift control problem was recently also solved using Reinforcement Learning (RL) (Cai et al., 2020). The authors adopted Soft Actor–Critic (SAC), the state-of-the-art model-free deep RL algorithm, to train a closed-loop drift controller. Although they show a satisfactory level of generalization (e.g., on various road structures, tire friction, and vehicle types), as it is an RL approach, no performance guarantees can be provided. Apart from that, it is still only a controller without any decision-making. An alternative approach for deciding when to drift is using Finite State Machines as presented by Acosta et al. (2019). However, this is also suboptimal and requires extensive engineering, and provides no guarantees on generalization to other scenarios. A more detailed overview of different approaches in the performance-driving domain is presented by Betz et al. (2022).

Driving on arbitrary circuit shapes (e.g., including S-like curves) generally requires to be able to generate trajectories for diverse curves, curves with variable curvature radius, and combinations of right and left curves. Optimal driving then consists of not only a single steady-state drifting maneuver but also close-to-straight driving and steady-state drifting in both directions (i.e., right and left). It is obvious that simple rules do not generalize and the question "*When* to go into *which mode* and for *how* to drive in it?" is an open problem. Solving this question can be considered a combinatorial optimization problem (i.e., NP-hard). Besides the combinatorial nature, to decide on the discrete mode (e.g., drifting, close-to-straight driving), the algorithm requires information on the feasibility of that mode (i.e does there exist a collision-free motion trajectory for that mode). This in turn makes it a class of Integrated Task and Motion Planning (TAMP) problems (hybrid systems), that are shown to be hard to solve optimally in a real-time (Garrett et al., 2021).

As deciding on the modes and generating references for full-circuit driving problems can be considered a combinatorial optimization problem, heuristic search methods can be a well-suited approach. In this paper, we present a novel A* search-based approach for generating "trackable" vehicle driving trajectories that exploit full vehicle dynamics. The presented A* search-based planner is a modified version of the one presented by Ajanovic et al. (2018), where a rather simple vehicle model was used to generate vehicle trajectories in complex urban driving scenarios. Here, a complex model of vehicle dynamics is used and motion primitives are generated using two different local approximations of the vehicle based on their feasibility (e.g., using the mode-feasibility-map). A bicycle model is used for maneuvers with small side-slip angle values (e.g., entry and exit maneuvers and close-to-straight driving) and an approximation of the full nonlinear vehicle model based on steady-state drifting is used for cornering maneuvers. The space of the possible trajectories is explored in an automated way, by systematically sampling different combinations of motion primitives, guided by a heuristic search. By using different locally approximated models for motion primitives generation, our approach can generate trajectories for arbitrary roads and assign appropriate modes for different segments, effectively overcoming the limitations of state-of-the-art approaches. A limited version of this work has been presented at the International Symposium on Dynamics of Vehicles on Roads and Tracks (Ajanovic et al., 2019). This work focuses more on AI principles rather than vehicle dynamics and it is a significantly extended version of the work, with more mature and well-elaborated algorithms and methods, more realistic experimentation and benchmarking to other approaches.

Some aspects of our work are related to other well-known approaches in the literature. Firstly, our approach for approximation of the full nonlinear vehicle model in steady-state cornering maneuvers is related to the closed-loop prediction approach (CL-RRT) of Kuwata

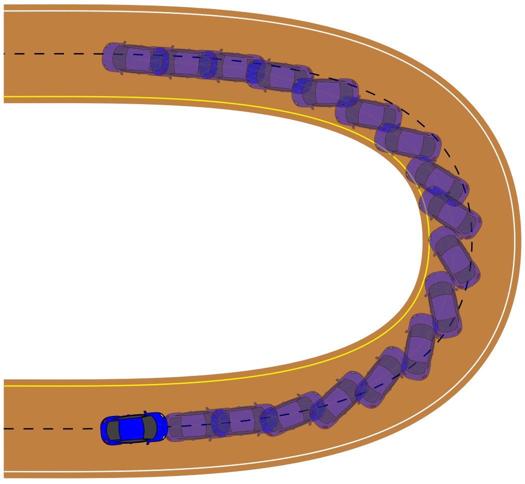


Fig. 1. Agile automated driving on a slippery road.

et al. (2008). They use a closed-loop system model to generate motion primitives when open-loop dynamics are unstable and the exploration by variations of the inputs in the open-loop dynamics becomes inefficient. In our approach, to avoid the same problem, we generate steady-state cornering motion primitives, which are assumed to be executable via closed-loop control, such as ones of Regolin et al. (2018) or Goh et al. (2019). However, the difference is that we collect extensive number of different equilibrium states in a manifold prior to planning and then sample directly from that manifold during the planning. Secondly, it is worth mentioning that besides motion planning for agile automated driving (as shown here), search-based planning was used for automated driving in unstructured environments (Montemerlo et al., 2008; Adabala and Ajanovic, 2020) and urban automated driving (Ajanovic et al., 2018). As well as for other challenging problems like planning footsteps for humanoid robots (Ranganeni et al., 2020), robot manipulation (Mandalika et al., 2018), underwater vehicles (Youakim et al., 2020) and the aggressive flying of UAVs (Liu et al., 2018). Different from these works, we introduce mode-feasibility-map that enables us to utilize multiple local model approximations and improve planning performance. Finally, our mode-feasibility-map resembles the initiation set of options framework (Sutton et al., 1999) in hierarchical reinforcement learning or preconditions in PDDL (McDermott et al., 1998) and STRIPS (Fikes and Nilsson, 1971) planning languages. Different from planning languages, we deal also with continuous dynamics instead of only logic. And different from Hierarchical RL, we employ mode-feasibility-map with planning algorithms.

The paper is structured as follows. Section 2 provides the problem formulation including necessary models as well as performance criteria and formal problem definition. In Section 3, the framework for motion planning and the approach for the generation of motion primitives are presented. Experimentation, including details on implementation and simulation study, is presented in Section 4. And finally, the conclusions and outlook of the work are presented in Section 5.

2. Problem formulation

The goal of this work is to develop a decision-making and control method that achieves a minimum lap time driving on an empty track in low friction conditions, e.g., gravel road. We assume that the vehicle is equipped with a map of the road and a localization system. Therefore, the vehicle has the information about the road ahead, as well as left/right boundaries and exact position and orientation. Moreover, the full vehicle state feedback information is assumed to be available. In particular, besides dynamic states, the low-level controller for state tracking requires measurements and estimations of several quantities, including wheel forces and wheel slips, both longitudinal and lateral (Regolin et al., 2019). Finally, the combined longitudinal/lateral tire–road contact force characteristics are assumed to be known and constant. The road, on the other hand, is assumed to be empty, flat, with static road–tire characteristic and can have arbitrary shape. The typical scenario could be driving in sharp curves (e.g., with radius 15 m) and entering high side-slip angle states (drifting) as shown in Fig. 1. As it can be seen, to achieve minimum lap-time, the vehicle has to go into drifting mode. In a such mode, the vehicle velocity and the longitudinal axis of the vehicle are not aligned, so the vehicle practically slides laterally.

Assumptions and requirements for this problem are summarized as follows.

Assumptions:

A1 The AV drives on a slippery road (e.g., gravel road) with known, constant road–tire characteristic.

A2 The AV is equipped with a map of the road and a state estimation (including localization) system.

A3 The AV drives on an empty track.

Requirements:

R1 The AV should drive safely on the road while aiming for the minimum lap time.

R2 The AV should be capable to drive in arbitrary planar road geometry (e.g., varying curvature radius, mixed right and left curves) without need for adjustments.

To properly define this problem, several aspects have to be defined including vehicle dynamical model, driveable road and performance criteria.

2.1. Vehicle dynamical model

Vehicle trajectories are generated by concatenating smaller segments of trajectories, the so-called "motion primitives" which are generated based on the model for the vehicle planar motion. Therefore, appropriate vehicle models are essential for the feasibility of the final trajectories. Modeling the planar motion of the vehicle for agile automated driving is very challenging as there are multiple aspects to be considered, including longitudinal, lateral, and yaw dynamics, tire–road forces, load transfer, etc. For motions that do not push the vehicle to the limits of handling, several simplifications can be used to effectively reduce the problem's complexity. For example, for small side-slip angle motions, a linearized model is accurate enough to be used. On the other hand, for motions that exploit full vehicle dynamics, more detailed vehicle models are required. A deep insight into vehicle dynamics modeling is presented by Tavernini et al. (2013). In this work, we present vehicle dynamics only at the level necessary to introduce the Equilibrium State Manifold (ESM) concept, which is used for the generation of motion primitives in Section 3.

To describe appropriately vehicle planar motion, the dynamic vehicle model is comprised of six state variables $[x, y, \psi, v, \beta, \dot{\psi}]^T$, where x, y, and ψ represent kinematic states (position coordinates and yaw angle), and v, β, and $\dot{\psi}$ are vehicle velocity, side-slip angle and rate of change of yaw angle respectively (as seen in Fig. 2). The evolution of x and y is given by the kinematic relations

$$\begin{cases} \dot{x} = v \cdot cos(\psi + \beta) \\ \dot{y} = v \cdot sin(\psi + \beta) \end{cases} \tag{1}$$

whereas the evolution of v, β, $\dot{\psi}$ are governed by the higher order vehicle dynamic model. In regular driving situations (i.e., for small values of β and $\dot{\psi}$), the vehicle dynamics identified by (1) is mostly determined by v and $\dot{\psi}$. Therefore, motion can be planned by means of linearized vehicle models. For cornering maneuvers, especially on slippery surfaces, such a solution is not suitable anymore, due to the effect of β in (1) as well as the complexity of the model that accurately

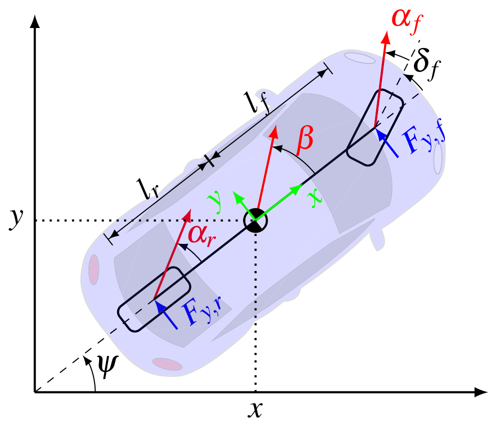


Fig. 2. Bicycle vehicle model.

describes the evolution of β itself, and might require a full nonlinear vehicle model.

For the cornering maneuvers, we use the full-vehicle nonlinear model, where longitudinal and lateral tire–road forces F_x, F_y for each (front and rear) axle are obtained from the normal forces F_z and the combined longitudinal-lateral friction model (Pacejka, 2012)

$$F_{x,i} = F_{z,i}\mu_x(\lambda_i,\alpha_i), \quad F_{y,i} = F_{z,i}\mu_y(\lambda_i,\alpha_i), \tag{2}$$

where μ is the friction coefficient, λ and α the longitudinal and lateral slips respectively. The front longitudinal force $F_{x,f}$ is zero due to the rear-wheel drive (RWD) configuration.

The nonlinear friction functions take the form of the *Magic Formula* (MF) tire friction model, with an isotropic friction model being used for simplicity (Pacejka, 2012). This requires the computation of the theoretical slip quantities (σ_j, $j \in \{x,y\}$), which can be obtained from α, λ as follows

$$\sigma_x = \frac{\lambda}{1+\lambda}, \quad \sigma_y = \frac{\tan\alpha}{1+\lambda}, \quad \sigma = \sqrt{\sigma_x^2+\sigma_y^2}. \tag{3}$$

Then, the one-directional friction coefficients are given by

$$\mu_i = \frac{\sigma_i}{\sigma} D \sin[C_\lambda \arctan\{\sigma B - E(\sigma B - \arctan\sigma B)\}], \tag{4}$$

for $i \in \{x,y\}$, with $B = 1.5289$, $C = 1.0901$, $D = 0.6$, $E = -0.95084$ being the Pacejka parameters corresponding to gravel.

The vehicle responses can be obtained from the following system of nonlinear equations, which considers the lateral, longitudinal, and rotating balance equilibrium equations around the vehicle center of gravity (COG):

$$\begin{cases} \dot{v} = \dot{\psi} v \beta + \frac{F_{x,r}}{m} \\ mv(\dot{\beta}+\dot{\psi}) = F_{y,f} + F_{y,r} \\ J_z\ddot{\psi} = l_f F_{y,f} - l_r F_{y,r} \end{cases} \tag{5}$$

where J_z is the vehicle inertia around the z-axis, m is the vehicle mass, and l_f, and l_r are the distances of the vehicle COG from the front and rear axles respectively. In addition, also longitudinal weight transfer is considered.

A major limitation that stems from using a full nonlinear vehicle model is the so-called "curse of dimensionality" (Bellman and Dreyfus, 1962), which causes a computational explosion when using higher dimensional models in numerical algorithms. In fact, if we use the full nonlinear vehicle model (6-dimensional) to generate motion primitives, the computational burden increases excessively (as we need exponentially many samples to properly sample it), thus making it a non-viable option for real-time implementation. Therefore, lower dimensional models are preferred regarding computational requirements. To overcome the unnecessary increase in computation requirements while maintaining the accuracy of the model, we develop and employ two local model approximations with respective domains of applicability. These are:

- Equilibrium States Manifold ($\mathcal{M}_{\mathrm{ESM}}$): a convenient approximation of the full nonlinear model, based on the steady-state drifting phenomenon, applicable during cornering (Section 2.1.1).
- Semi-linearized bicycle model approximation ($\mathcal{M}_{\mathrm{lin}}$): applicable in straight-driving/mild-turning scenarios (Section 2.1.2).

2.1.1. Equilibrium states manifold

As previously mentioned, "donut drifting" is well known in practice (i.e., exploited by drivers) and investigated in the research (also known as so-called steady-state drifting). We utilize this phenomenon and expand the concept by collecting extensive number of (desirably all) feasible steady-states in a manifold that we call Equilibrium States Manifold (ESM). To obtain such a manifold, we perform offline numeric computations on the full nonlinear vehicle model (and in other case extensive simulations), and collect the feasible steady-state solutions of the vehicle cornering at different curvature radii (Velenis et al., 2011). These solutions include the vehicle control inputs (steering wheel angle and rear wheels slip), as well as vehicle states v, β, $\dot{\psi}$. Assuming a RWD drivetrain configuration, and given different sets of values of the constant control inputs (steering wheel angle δ, driving wheels slip λ), multiple equilibrium points $s_{ESM} = [v_{ss}, \beta_{ss}, \dot{\psi}_{ss}]^T$ can be computed for a given constant curvature radii R_c, by considering the uniform circular-motion relation $\dot{\psi} = \frac{v}{R_c}$, and imposing the steady-state condition (6) in the vehicle model (5).

$$\dot{v} = \dot{\beta} = \ddot{\psi} = 0 \tag{6}$$

A race track is composed of different sections, with varying curvature radii. Therefore, in order to model steady-state drifting with different radius, we need to compute different equilibrium points. For this reason, the sets $\mathbf{s}_{\mathrm{ESM}}(R_{c_i})$, for different curvature radii $i \in \{1,\dots,r\}$ are computed and then interpolated into a map $v = f(\beta,\dot{\psi})$, which represents the ESM ($\mathcal{M}_{\mathrm{ESM}}$), as follows.

$$\mathcal{M}_{\mathrm{ESM}} = \{(v,\beta,\dot{\psi}) \in \mathbb{R}^3 \mid \dot{v} = \dot{\beta} = \ddot{\psi} = 0\}. \tag{7}$$

ESM ($\mathcal{M}_{\mathrm{ESM}}$) is later used to generate steady-state motion primitives by sampling different states s_{ESM}. The same procedure is applied for δ, and λ. In Fig. 3, these sets $\mathbf{s}_{\mathrm{ESM}}(R_{c_i})$ are visualized in the 3-dimensional state-space for varying R_c together with the final interpolated ESM ($\mathcal{M}_{\mathrm{ESM}}$). The corresponding surfaces, generated for δ and λ are displayed in Fig. 4.

Let us assume that the tire–road contact model and the vehicle dynamics model (2)–(5) describe accurately the cornering maneuver dynamics and that a path with curvature radius R_c is given, for which at least one reference state $s_{ESM}(R_c)$ exists. Then, if a locally stable feedback controller for the tracking of the state is designed, such path is feasible and can be tracked with appropriate velocity and side-slip angle, given an initial condition close enough to the target state.

2.1.2. Semi-linearized bicycle model approximation

When driving conditions are close enough to the origin of the $\beta-\dot{\psi}$ plane (e.g., close-to-straight driving), a nonlinear bicycle model can be simplified and we can use the semi-linearized bicycle model (Genta, 1997). Therefore, the forces in (5) can be replaced with their linearized approximations as

$$F_{y,f} = -C_f(\beta + l_f\frac{\dot{\psi}}{v} - \delta) \tag{8a}$$

$$F_{y,r} = -C_r(\beta - l_r\frac{\dot{\psi}}{v}) \tag{8b}$$

$$F_{x,r} = -C_x\lambda. \tag{8c}$$

In (8), the longitudinal and lateral stiffness coefficients C_x, C_f, C_r are consistent (linearized approximation) with the full characteristics given by (4) on a given domain. This model ($\mathcal{M}_{\mathrm{lin}}$) is valid for range defined by $|\beta| < \beta_{\mathrm{lin}}$ and $|\dot{\psi}| < \dot{\psi}_{\mathrm{lin}}$ where linearization error is negligible.

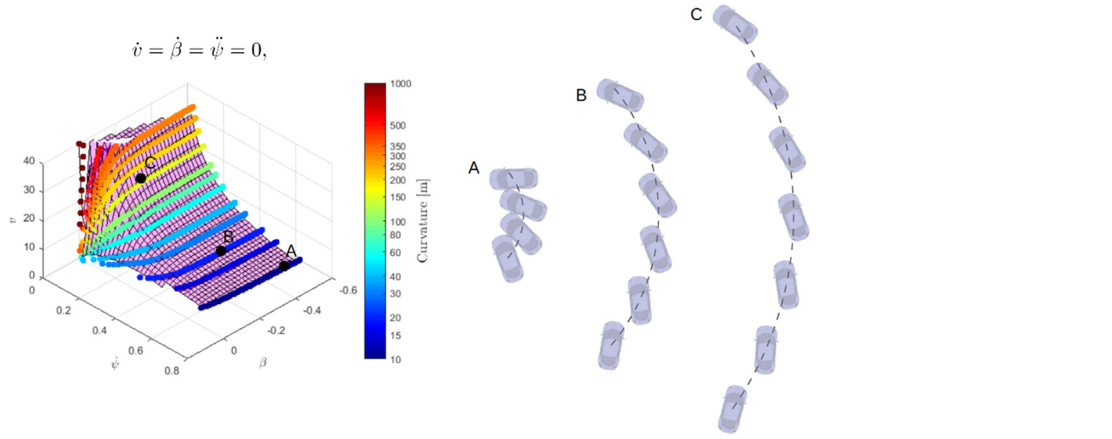


Fig. 3. On the left, equilibrium points sets $\mathbf{s}_{\mathrm{ESM}}$ (and linear interpolation $\mathcal{M}_{\mathrm{ESM}}$) in the $v \times \beta \times \dot{\psi}$ space for counter-clockwise cornering maneuvers with different curvature radii R_c. On the right, three segments of motions corresponding to three different states s_{ESM} on the Equilibrium State Manifold $\mathcal{M}_{\mathrm{ESM}}$ (A, B and C).

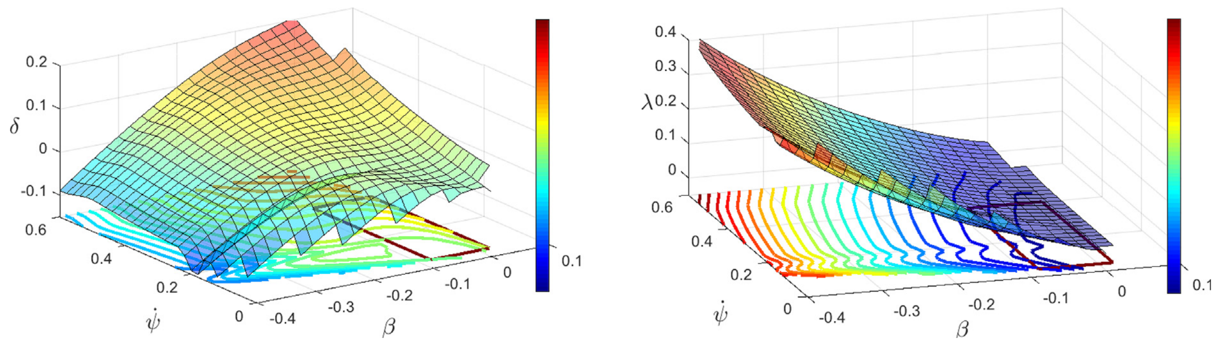


Fig. 4. ESM for the inputs δ (left) and λ (right), counter-clockwise maneuvers. The highlighted portion of the $\beta \times \dot{\psi}$ plane corresponds to the one in which the bicycle model representation is considered valid. The intervals of values $[\delta_{\min}, \delta_{\max}]$, $[\lambda_{\min}, \lambda_{\max}]$ are used for the bicycle-model expansion explained in Section 3.2.

2.2. Driveable road

We consider that the vehicle can drive only on the road $\mathcal{R}$. The states outside of the road are considered to be non-driveable states and they are treated as obstacles ($\mathcal{O}$) by the motion planning algorithm. Therefore, the driveability of the trajectory generated based on the vehicle model can be validated based on vehicle coordinates x, y, and yaw angle ψ only (no need to consider higher dynamical states). As road geometries vary a lot, they can introduce unnecessary complications for motion planning to generate a trajectory that keeps the vehicle on the road. To simplify planning, the driveable road is modeled using a Frenet frame (Werling et al., 2012). Instead of using x and y coordinates, in the Frenet frame, one dimension represents the distance traveled along the road s, and the other represents the deviation d from the road centerline. By using the Frenet frame, some operations become trivial. For example, to determine whether the vehicle is on the road, it is sufficient to check if the lateral deviation d in the Frenet frame is exceeding half of the road width $w_{\mathrm{road}}/2$.

The Frenet frame also ensures that the planning procedure remains the same for each segment of the road. It is important to note that operations in the Frenet frame are used only for trajectory evaluation during planning (e.g., distance traveled, collision checking if the vehicle is on the road, etc.) and grid forming for underlining data structure in planning. On the other hand, the vehicle dynamic model in the Cartesian coordinate system is still used for motion primitive generation. Therefore, we effectively avoid problems of Frenet frame like shown by Li et al. (2022). Efficient transformations between Frenet and Cartesian frames are necessary as they are used frequently (for every explored node) in each planning step. Fig. 5 illustrates the procedure of this transformation. Road geometry from the Cartesian coordinate system (left) is represented as a straight road in the Frenet frame (right). Additionally, to ensure all parts of the vehicle are on the road, the vehicle can be represented using multiple circles as shown by Ziegler and Stiller (2010). Ensuring all circles are on the road ensures the vehicle is on the road as well. This is ensured by checking the following condition:

$$|d_i| \leq \frac{w_{\mathrm{road}}}{2} - r_i \tag{9}$$

for each circle i and respective lateral deviation d_i and circle radius r_i.

2.3. Performance criteria

The goal of the planner is to minimize the time T necessary to drive the full lap. In the distance-based formulation, the criteria can be formulated as follows.

$$T = \int_0^{s_{\mathrm{G}}} \frac{ds}{v(s) \cdot \cos\big(\psi(s) + \beta(s) - \psi_{\mathrm{road}}(s)\big)}. \tag{10}$$

where term $\psi(s) + \beta(s) - \psi_{\mathrm{road}}(s)$ represents the angle between vehicle velocity and the road tangent. So, the whole determinant represents the component of the velocity along the road.

As can be seen, this equation uses the distance as integral bound variables (0 and s_{G}), as it is easier to relate it to the lap start and lap end. The cost function formulated like this can be used to find the global optimal solution.

However, in MPC, with a fixed time horizon, another formulation can be used to achieve the same effect. Since the goal is to minimize lap time T and the planning time horizon is fixed, equivalent behavior can be achieved by maximizing the distance traveled along the road for a defined time horizon. The criteria can be evaluated simply by considering the first coordinate in the Frenet frame, distance along

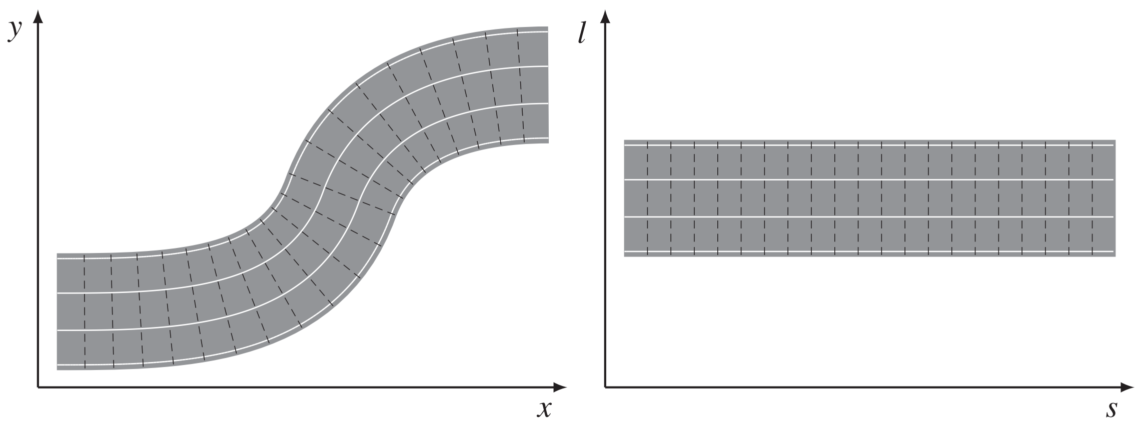


Fig. 5. Curvilinear road representation (Cartesian and Frenet-frame).

the path s, which is trivial. In the Cartesian frame, this would be equivalently represented as in (11).

$$J_{\max} = \max_{u(\cdot)} \int_0^{t_{\text{hor}}} v(t) \cdot \cos(\psi(t) + \beta(t) - \psi_{\text{road}}(s))dt. \quad (11)$$

2.4. Formal problem definition

Finally, based on the presented vehicle model and the driveable road we can define the search space that considers kinodynamic constraints imposed by vehicle dynamics:

$$\mathcal{X} = \left\{ \mathbf{x} \equiv \left[s, d, \psi, v, \beta, \dot{\psi}, t\right]^T \mid (s,d) \in \mathcal{R}, \psi \in \left[0, 2\pi\right], v \in \left[0, v_{\max}\right], \right.$$
$$\left. \beta \in \left[\beta_{\min}, \beta_{\max}\right], \dot{\psi} \in \left[\dot{\psi}_{\min}, \dot{\psi}_{\max}\right] \right\}. \quad (12)$$

As planning is executed in a moving horizon fashion with a fixed time horizon, the goal region is defined as:

$$\mathcal{X}_G = \{\mathbf{x} \mid \mathbf{x} \in \mathcal{X}, t \geq T_{\text{hor}}\}. \quad (13)$$

Agile automated driving motion planning problem can be formally formulated as follows.

Given:

- the **search space:** $\mathcal{X}$, $(s \times l \times \psi \times v \times \beta \times \dot{\psi} \times t)$,
- the **vehicle model:** Equilibrium State Manifold (Section 2.1.1), semi-linearized bicycle model (Section 2.1.2),
- **constraints:**
  - **internal:** acceleration and velocity , surface (Section 2.1.1), steering and slip (Section 2.1.2),
  - **external:** vehicle is on the road $|l| \leq \frac{w_{\text{road}}}{2}$ with is slightly more complex extension (Ziegler and Stiller, 2010) for the full vehicle geometry,
- **objective:** minimum lap-time (Section 2.3),
- **a query:** initial state $\mathbf{x}_0$ and the final state region $\mathcal{X}_G$.

Compute a continuous path $\gamma(\cdot)$ that moves the vehicle from the initial state to the goal region while satisfying all the constraints ($\gamma : [0,1] \mapsto \mathcal{X}_{\text{free}}$ such that $\tau(0) = \mathbf{x}_0$, $\tau(1) \in \mathcal{X}_G$) and minimizing the objective.

3. Task and motion planning approach

In this section, we present our search-based task and motion planning framework (SBMP), used for the generation of the driving trajectory. First, we describe some general aspects of the SBMP framework, followed by the clarification of individual components like node expansion and heuristic function, etc. as it can be seen on Fig. 6.

3.1. SBMP framework

The proposed Task and Motion planning framework is based on the A* search method (Hart et al., 1968), guided by a heuristic function in an MPC-like replanning scheme. After each time interval T_{rep}, replanning is triggered from the current vehicle state $\mathbf{x}$, together with information about the driveable road ahead $\mathcal{R}$. The feasible vehicle trajectories are constructed by concatenating smaller segments of trajectories, the so-called *motion primitives* (Frazzoli et al., 2002). The space of the possible trajectories is explored by sampling different combinations of motion primitives in a systematic way, guided by a heuristic search. Motion primitives are generated using two different locally approximated vehicle models. A semi-linearized bicycle model ($\mathcal{M}_{\text{lin}}$) is used for small side-slip angle operations (e.g., curve entry and exit maneuvers and close-to-straight driving) and an approximation of the full nonlinear vehicle model ($\mathcal{M}_{\text{ESM}}$) for steady-state cornering maneuvers.

The trajectory is constructed by a grid-like search using an A*-like algorithm shown in Algorithm 1. The grid is constructed via equidistant discretization of the state variables $\mathbf{x}$ in all 7 dimensions. It is important to highlight that the full graph is not constructed in advance, but is built iteratively as the search progresses. In this way, only a small portion of the search space is explored and saved in memory. As we search in the continuous search space $\mathcal{X}$ and expand nodes by sampling multiple motion primitives that generally do not end exactly at grid points. Rounding continuous state to the grid would introduce accumulation of the rounding error. Therefore, an adaptation of the hybrid A* approach (Montemerlo et al., 2008) is used for the search. Hybrid A* also uses the grid, but keeps continuous values as well, without rounding it to the grid. When a node is expanded, motion primitives are initiated from the exact continuous state, thus preventing the accumulation of rounding errors. Additionally, keeping only one node in each grid box prunes unnecessary trajectories making it more efficient than purely sampling-based methods.

Each *node* n contains 20 variables: 6 indexes (representing the grid box) - one for each state in $\mathbf{x}$ ($n.x_k$, $n.y_k$, $n.\psi_k$, etc.), 6 indexes for the *parent node* (used to reconstruct the solution trajectory at the end of search), six continuous remainders from the discretization of states (used for the initialization of motion primitives) - one for each state in $\mathbf{x}$ ($n.x_r$, $n.y_r$, $n.\psi_r$, etc.), the exact *cost-to-come* to the node ($n.g$), and the estimated total cost of traveling from the initial node to the goal region ($n.f$). The value $n.f$ is computed as $n.g + h(n)$, where $h(n)$ is the heuristic function.

Starting from the *initial node* n_{I} (i.e., representing the initial state), chosen as the first *current node* n. At each iteration, successor nodes are generated in the function `Expand` by expanding the *current node* n using motion primitives that are dynamically feasible from that node. The end state from each collision-free motion primitive is represented

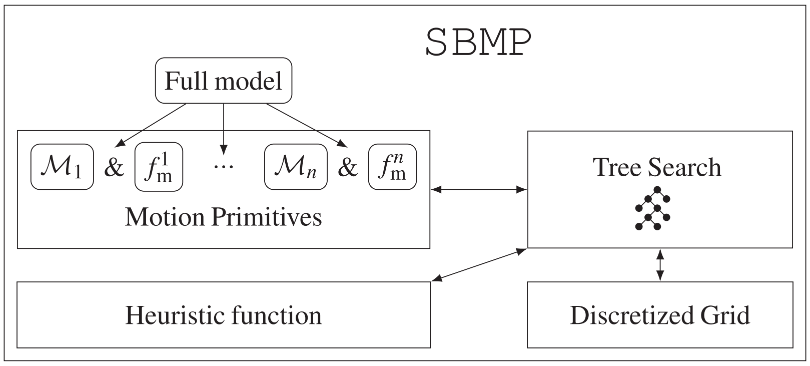


Fig. 6. SBMP framework.

Algorithm 1: `Search`: search-based plan for a horizon

```
input : $n_I$, $(\mathcal{R})$, $\mathcal{M}_{lin}$, $\mathcal{M}_{ESM}$, $f_m$, $h(n)$
1  begin
2      $n \leftarrow n_I$                              // initialization
3      CLOSED $\leftarrow \emptyset$
4      OPEN $\leftarrow n$
5      while $n.k \leq k_{hor}$ and OPEN $\neq \emptyset$ and OPEN.$size() \leq N_{timeout}$ do
6          $n \leftarrow$ Select(OPEN)
7          OPEN $\leftarrow$ OPEN $\setminus n$
8          CLOSED $\leftarrow$ CLOSED $\cup$ $n$
9          $(\mathbf{n}', \mathbf{n}'_C) \leftarrow$ Expand$(n, \mathcal{M}_{lin}, \mathcal{M}_{ESM}, h(n))$
10         CLOSED $\leftarrow$ CLOSED $\cup$ $\mathbf{n}'_C$
11         OPEN $\leftarrow$ OPEN $\cup$ $\mathbf{n}'$
12     return GetTraj $(n)$      // reconstruct trajectory
```

with one reachable *child node*. All *child nodes* $\mathbf{n}'$ are processed and some are added to the OPEN list. If the *child node* is already in the OPEN list, and the new *child node* has a lower cost, the parent of that node is updated, otherwise, it is ignored. From the OPEN list, at every iteration, the node with the lowest cost is chosen to be the next *current node* (in the function `Select`), and the procedure is repeated until the horizon is reached, the whole graph is explored or the computation time limit for planning is reached. At the end of the planning, the node closest to the horizon is used to reconstruct the trajectory.

The planning clearly requires processing time. The compensation for the planning time can be achieved by introducing T_{plan}, a guaranteed upper bound on planning time. The planning is then initiated from the state $\mathbf{x}(t + T_{plan})$, at which the vehicle would be after the T_{plan} time. In this way, the old trajectory is executed while the new one is planned. Thus, the new trajectory is already planned at $t + T_{plan}$. This approach has been widely used in MP for automated vehicles (Ziegler et al., 2014).

3.2. Node expansion and motion primitives

To build trajectories iteratively, at each iteration of the search, current node n is expanded, and *child nodes* $\mathbf{n}'$ are generated using motion primitives. From each node, n, only dynamically reachable and collision-free *child nodes* $\mathbf{n}'$ are generated. Each generated child node n' in $\mathbf{n}'$ represents the end state from one motion primitive trajectory. As mentioned before, we employ two types of motion primitives (from two locally approximated models), depending on the *mode-feasibility-map* f_m. The first type, based on Equilibrium State Manifold $\mathcal{M}_{ESM}$, is for steady-state drifting during cornering. The second, based on the semi-linearized bicycle model $\mathcal{M}_{lin}$, is for close-to-straight driving.

Fig. 7 illustrates motion primitives for two presented models. On the left, motion primitives are generated using a semi-linearized bicycle model $\mathcal{M}_{lin}$ with 2 variations in the rear wheels slip λ and 3 variations

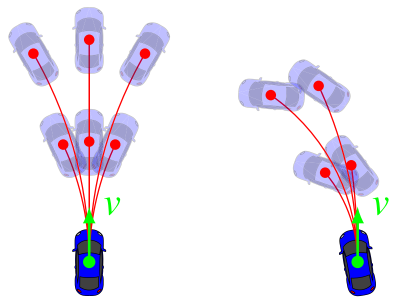


Fig. 7. Motion primitives for agile driving using semi-linearized bicycle model $\mathcal{M}_{lin}$ (left) and ESM $\mathcal{M}_{ESM}$ (right).

in steering wheel angle δ. In total 6 motion primitives are generated. On the left, 4 motion primitives are generated by sampling in ESM $\mathcal{M}_{ESM}$. In practice, many more motion primitives are generated, up to about 100 successor nodes for each expanded node and around 2000 in total for the whole planning step. The complete procedure for the `Expand` procedure is described in Algorithm 2.

Algorithm 2: `Expand`: generating child nodes based on motion primitives

```
input : $n$, $\mathcal{M}_{lin}$, $\mathcal{M}_{ESM}$, $f_m$, $h(n)$
1  begin
2      $\mathbf{n}' \leftarrow \emptyset$
3      $\mathbf{n}'_C \leftarrow \emptyset$
           // check applicability based on $\beta - \psi$ map, Fig. 8
4      if $n \in f_m^{ESM}$ then
5          $\mathbf{n}' \leftarrow$ Sample$(n, \mathcal{M}_{ESM})$
6      if $n \in f_m^{lin}$ then
7          $\mathbf{n}' \leftarrow \mathbf{n}' \cup$ Sample$(n, \mathcal{M}_{lin})$
8      $\mathbf{n}'_C \leftarrow (\mathbf{n}' \mid \mathbf{n}' \notin \mathcal{R})$        // collision checking
9      $\mathbf{n}' \leftarrow \mathbf{n}' \setminus \mathbf{n}'_C$
10     return $(\mathbf{n}', \mathbf{n}'_C)$
```

3.2.1. Mode-feasibility-map

Mode-feasibility-map (f_m) represents domains of feasibility for each of the modes. It is a crucial component that enables solving complex continuous problems as a TAMP and it offers an elegant solution to enable the use of multiple modes or approximated models. Depending on the initial state $\mathbf{x}$, each of the modes might be feasible or infeasible. Mode-feasibility-map resembles the initiation set of options framework (Sutton et al., 1999) in hierarchical reinforcement learning or preconditions in PDDL (McDermott et al., 1998) and STRIPS (Fikes and Nilsson, 1971) planning languages.

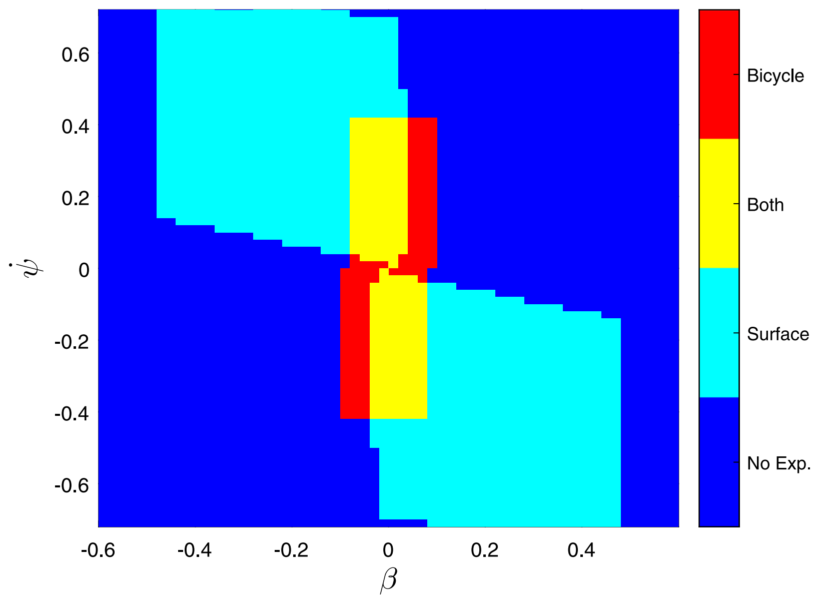


Fig. 8. Mode-feasibility-map f_m, representing feasibly of expansion modes depending on initial states in $\beta \times \dot{\psi}$.

For our agile driving problem, there are two distinct modes. These are steady-state drifting and close-to-straight driving, as described below. Mode feasibility, in this case, depends only on two states, side-slip angle β and yaw rate $\dot{\psi}$, as in (14).

$$f_m(\mathbf{x}) = f_m(\beta, \dot{\psi}) \tag{14}$$

As can be seen in Fig. 8, there are different regions of the $\beta \times \dot{\psi}$ plane. Steady-state drifting is feasible on the regions where Equilibrium State Manifold $\mathcal{M}_{\text{ESM}}$ is defined. As it can be seen Equilibrium State Manifold $\mathcal{M}_{\text{ESM}}$ is symmetric around the origin, as we extended it for drifting in both directions, clockwise and counter-clockwise. On the other hand, when initial state $\mathbf{x}$ is close enough to the origin of the $\beta \times \dot{\psi}$ plane, i.e., for $|\beta| < \beta_{\text{lin}}$ and $|\dot{\psi}| < \dot{\psi}_{\text{lin}}$, we can employ close-to-straight driving and use $\mathcal{M}_{\text{lin}}$ to generate motion primitives. As it can be seen, these two modes complement each other allowing a smooth transition between drifting from one direction to another. Each of the modes is used for generating motion primitives and respective child nodes whenever it is feasible.

3.2.2. *Steady-state drifting mode*

During cornering, motion primitives are generated based on the Equilibrium State Manifold $\mathcal{M}_{\text{ESM}}$, generated offline (as it is explained in Section 2.1.1), consisting of states $s_{ESM} = (v, \beta, \dot{\psi})$ each representing feasible solution for continuous drifting with different radii ("donut drifting"). A race track is composed of different sections, with varying curvature radii (as well as straight segments). Therefore, a continuous transition between different steady-states is desired. Based on the current node n (respective state $\mathbf{x}$), respective steady-state s_{ESM} on ESM ($\mathcal{M}_{\text{ESM}}$) is obtained by projecting onto the manifold $\mathcal{M}_{\text{ESM}}$. From ESM ($\mathcal{M}_{\text{ESM}}$) several reachable steady-states are sampled in the neighborhood of s_{ESM} and kinematic model (1) is used to simulate the evolution of the additional states (x, y, ψ), assuming the linear transition between the current and sampled neighboring steady-states, effectively generating steady-state drifting motion primitives (with full state trajectories). The final state of each motion primitive is used to generate one child node n'. Neighboring steady-states are obtained by sampling the $\beta \times \dot{\psi}$ space around the current s_{ESM} (with values $(\beta_0, \dot{\psi}_0)$), with the density of the samples decreasing as the distance from the s_{ESM} increases (see Fig. 9). This approach of sampling in $\mathcal{M}_{\text{ESM}}$ effectively reduced the problem from sampling trajectories in 7-dimensional space to sampling the point in 2-dimensional space. For us, this pattern of sampling showed good results. However, different sampling patterns could be also employed. The number of samples is a hyperparameter, which impacts considerably the performance of the search. As the number of samples is increased smoother trajectories can be planned, but the branching factor of the tree increases so computation time increases exponentially. Therefore a fine trade-off between computation time and sufficient space exploration is needed.

Eq. (6), used as a condition to generate the ESM, assumes that the rates of change of states are equal to zero. However, we need to transit between close states in order to generate trajectories with varying curvature in order to keep the vehicle on the road. This implies that this constraint (Eq. (6)) must be "softened". Still, it is important to keep it low, so limits on relative change must be set such as shown in (15).

$$\Delta v \leq \Delta v_{\max}, \tag{15a}$$

$$\Delta \beta \leq \Delta \beta_{\max}, \tag{15b}$$

$$\Delta \dot{\psi} \leq \dot{\psi}_{\max}. \tag{15c}$$

The smaller the deviations are, the closer the trajectory is to the ESM, therefore the model is more accurate. In practice, these limits are obtained experimentally by increasing them and detecting when trajectories become infeasible.

In order to avoid generating and propagating an excessive amount of samples (to decrease the branching factor of the search), even before the nodes are checked for collision and removed, the following rules are considered:

- only equilibrium points defined within the surface in Fig. 3 are considered. This also means that the minimum reachable curvature radius is $R_{\text{c,min}} = 10$ m;
- the (small) portion of the curve such that $\beta \cdot \dot{\psi} > 0$ is neglected since equilibrium points in which β and $\dot{\psi}$ have the same sign are associated with low-velocity conditions;
- a maximum velocity deviation Δv between two successive nodes is defined, such that $\frac{\Delta v}{T_s} < a_{\max}$, where $a_{\max}$ is the estimated maximum deceleration allowed on the given road surface.

3.2.3. *Close-to-straight driving mode*

When initial state $\mathbf{x}$ is close enough to the origin of the $\beta - \dot{\psi}$ plane, i.e., for $|\beta| < \beta_{\text{lin}}$ and $|\dot{\psi}| < \dot{\psi}_{\text{lin}}$, motion primitives are also generated according to a semi-linearized bicycle model, where the forces in (5) are replaced with their linearized approximations $\mathcal{M}_{\text{lin}}$. In order to generate

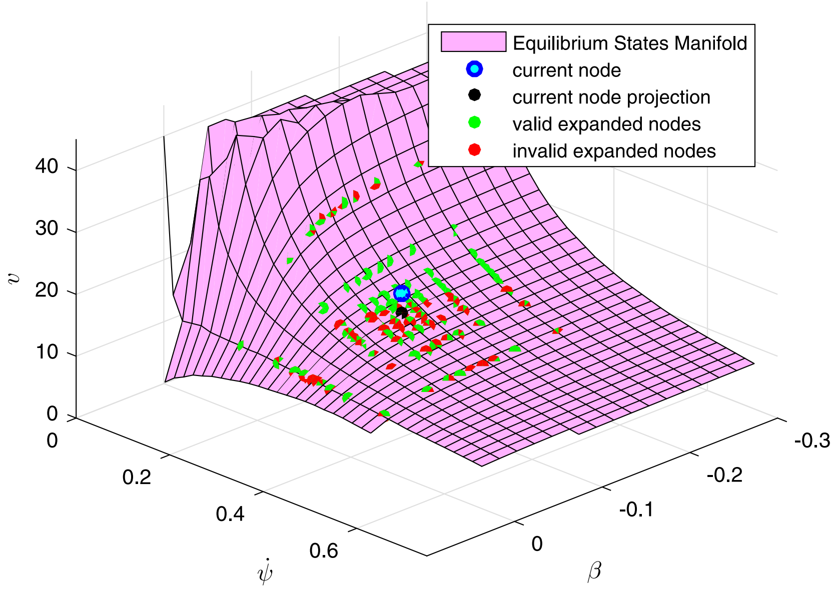


Fig. 9. Expanding parent node n to different child nodes $\mathbf{n}'$ by sampling on the ESM.

different motion primitives, inputs are varied such that different values for steering wheel angle δ and the rear wheels slip λ are equidistantly sampled within the ranges. Ranges are defined by $|\beta| < \beta_{\mathrm{lin}}$ and $|\psi| < \psi_{\mathrm{lin}}$ as in the equilibrium surfaces in Fig. 4. In this way, multiple motion primitives are generated with different end velocities and turning radii.

Finally, all expanded nodes are checked for a collision based on (9) and (15), and nodes that are in a collision are removed, leaving the remaining ones feasible regarding vehicle dynamics and driveability.

3.3. Heuristic function

The heuristic function $h(n)$ is used to guide the search. It estimates the cost needed to travel from some node n to the goal state (*cost-to-go*). As it is shown by Hart et al. (1968), if the heuristic function is underestimating the exact cost-to-go, the A* search provides the optimal trajectory. For the shortest path search, the usual heuristic function is the Euclidean distance. On the other hand, to find the minimum lap time, the heuristic should estimate the distance that the vehicle can travel from the current node during the defined time horizon. It is optimistic to assume that the vehicle accelerates (with maximum acceleration) in the direction of the road's central line until it reaches the maximum velocity, and then maintains it for the rest of the time horizon. Based on this velocity trajectory, the maximum travel distance can be computed and used as a heuristic.

In order to bias exploration towards the preferred motions and improve robustness, on the cost of sacrificing theoretical optimality, the heuristic function is augmented considering, among others:

- a "dynamic states evolution" cost, which helps limit the rate of change of the references v, β, $\dot{\psi}$, in order to obtain smooth trajectories and improve closed-loop state tracking;
- penalization for trajectories approaching the roadside;
- penalization of the nodes with fewer siblings, thus biasing the search to avoid regions where only a few trajectories are feasible.

3.4. Illustrative example

Constructed in this way, with presented components, SBMP can deal with nonlinear and hybrid vehicle models and plan for agile automated driving trajectories in a TAMP fashion. The method is generalizable to complex driving situations (arbitrary combinations of right and left curves and straight paths). Fig. 10 illustrates one such example. The

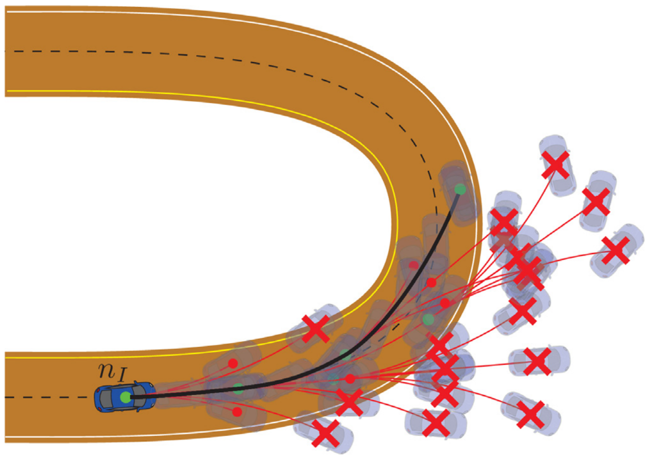


Fig. 10. Illustration of the principles of operation.

vehicle has to go into drifting mode to be able to drive through the sharp turn optimally. If drifting mode is not considered, the vehicle has to slow down significantly in order to stay on the road. As can be seen, many motion primitives lead the vehicle off the road and are therefore removed. Search is continued until some trajectory is found that keeps the vehicle on the road for the whole horizon.

4. Experimentation

The presented SBMP framework was adapted for the agile automated driving use case and implemented in the Matlab/SIMULINK environment. As mentioned before, the established approach in motion planning for automated driving is to start re-planning from some future state from the previous plan as long as there is no large tracking deviation from the planned motion (Ziegler et al., 2014). Therefore, we first verify planner performance assuming perfect actuation, i.e., the actual vehicle dynamical states/positions match the ones planned at the previous iteration. This is also important, as the focus of this work is on computationally efficient trajectory generation. Additionally, to prove that planned trajectories are feasible in the real system, we also show the performance of planned trajectory tracking in a closed

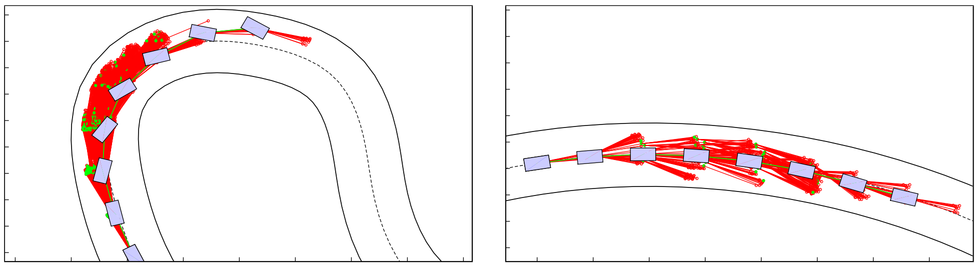


Fig. 11. Graphical representation of the trajectories exploration in U-turn (left) and wide turn (right).

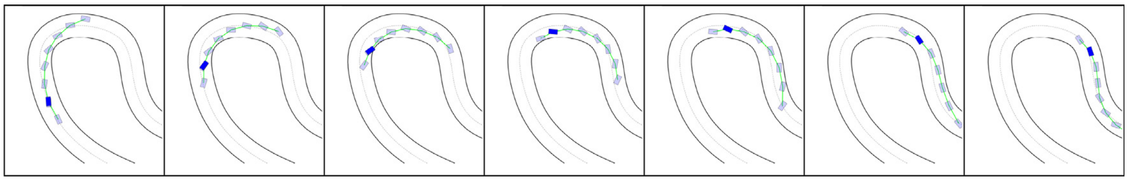


Fig. 12. U-turn maneuver: consecutive frames.

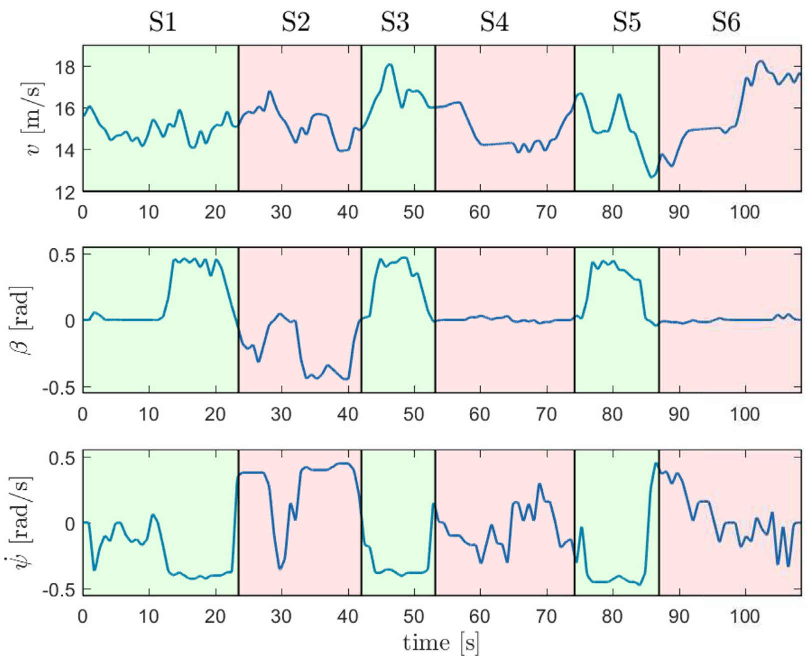


Fig. 13. Reference dynamical states over the full test circuit.

loop using controllers on the full nonlinear vehicle model. For verification purposes, an artificially mixed circuit was used, characterized by slippery conditions (gravel), which contains several road sections of varying curvature radii, as can be seen in Fig. 14. The proposed planner manages to find the appropriate vehicle trajectory for driving on the track. An example of the algorithm exploration behavior is shown in Fig. 11 in the case of a U-turn and of a wider curve. The explored branches are represented by the red links, and the closed nodes are marked as green. The light-blue car frames represent the optimal vehicle states (see Fig. 11).

In Fig. 12, several frames of the same maneuver are shown (the top left turn in the track illustrated in Fig. 14). From these, it is possible to get an insight into how the optimal trajectory is re-planned, at each iteration, based on the current position. Given the nature of the receding horizon approach, it is not guaranteed (nor preferred) that all or part of the previously computed trajectory are kept in the next iteration. In fact, while in the first step the trajectory approaches "dangerously" the side of the road, in the next two steps the trajectory is incrementally improved, thanks to the fact that the exploration of such a portion of the track is now being evaluated in earlier nodes.

The dynamical states, which represent the output of the trajectory generation, are depicted in Fig. 13. One can see how the generated references are varied smoothly, in particular in terms of v and β, which are the quantities characterized by slower actuation dynamics. Moreover, it is possible to distinguish clearly 4 intervals in which the optimal generated maneuver is a 'drift' one with $\beta > 0.4$ rad. These same intervals can be distinguished in Fig. 14, where the overall trajectory on the considered 10 m-wide track can be evaluated.

For validating the advantage of using multiple modes (i.e., drifting) we benchmark our approach to other state-of-the-art approaches that do not use drifting mode but can still drive full circle autonomously. Due to the low friction coefficient of the dirt road, we found other approaches like Liniger et al. (2015) and Li et al. (2022) difficult to adapt for these conditions and achieve the full circuit driving. After extensive unsuccessful trials, we used our planner with disabled drifting mode. We found that solution as well representative of these approaches as

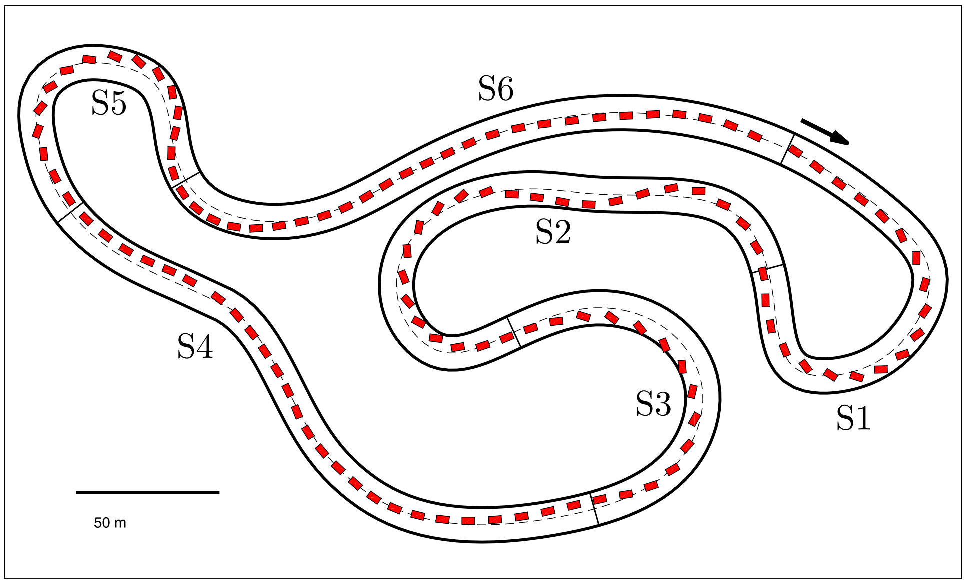


Fig. 14. Obtained driving trajectory over the full test circuit.

it uses the model with similar fidelity, but also has a defined domain of the applicability from mode-feasibility-map f_m. Mode-feasibility-map enables it to know the limits of the model and provide feasible trajectories also for low friction conditions. SBMP without drifting is achieving an average velocity of approx. 8 m/s, while the average velocity of SBMP with both modes is approx 15 m/s. This demonstrates the advantage of using drifting on low-friction surfaces and achieving minimum-time cornering.

4.1. Controllers

For the execution of the motion plans, in this work, we use two different controllers. One for each of the modes. For steady-state cornering mode, we use a drifting controller (Goh et al., 2019). And for close-to-straight driving, we use the path-following controller (Lu et al., 2018), that includes a Sliding Mode Controller for lateral motion and a simple PI longitudinal controller that minimizes the weighted sum of velocity and position-lag error. We switch between these two controllers based on the mode selected by the planner.

As can be seen in Fig. 15, controllers are robust enough and trajectories are feasible so the vehicle can drive through a very challenging curve. At around 4 s, mode is changed from close-to-straight driving to drifting. This is seen also on the side-slip angle β in Fig. 16. The drifting controller successfully overtakes the control and continues through the curve. Although controllers are not tracking perfectly reference states, the final diving line is closely following the reference.

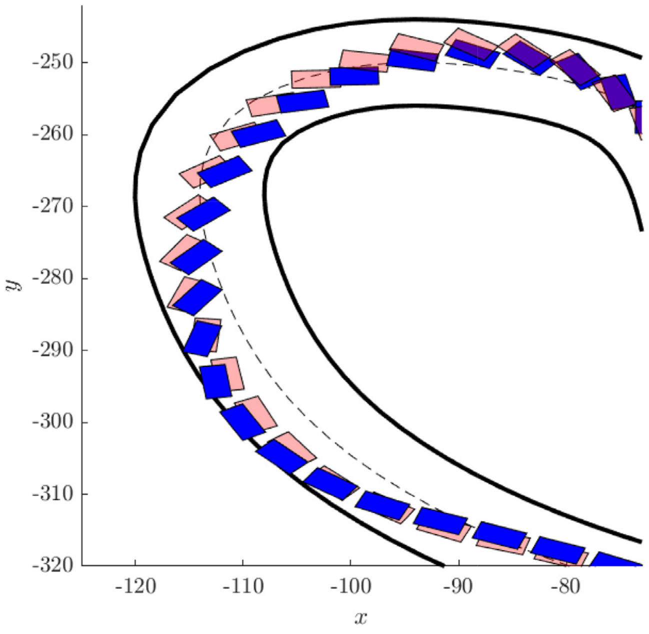


Fig. 15. Tracking reference trajectory using path tracking and drift controller.

4.2. Computational performance

It is well known that the computational complexity of the A* search depends on the quality of the heuristic function (Russell and Norvig, 2021). In the worst case, when the heuristic function is not informative at all, the algorithm behaves as an exhaustive search, with exponential time complexity in the depth (in the order of $O(b^d)$). Where b represents the branching factor and d represents the depth.

In our algorithm, the depth d represents horizon length. More precisely, the horizon time of MPC divided by the time-step length of motion primitives. On the other hand, the branching factor b represents the number of sampled motion primitives generated at each `Expand` step (in the order of 100). To achieve real-time algorithm performance, we utilized a hybrid A* approach that prunes generated motion primitives based on the discretized search space and practically reduces the branching factor. An additional advantageous feature of our approach is that we design the algorithm as an anytime algorithm with a timeout. If some planning instance is harder to solve, by limiting the number of nodes, we practically shorten the horizon so it can be solved faster. This provides a suboptimal solution, but the solution can be corrected again in the next re-planning step.

Besides theoretical computational complexity, practically, the usability of the algorithm very much depends on the constants in the complexity relation. Well-optimized implementation and appropriately tuned problem parameters can make it very practical. Besides the Matlab/SIMULINK implementation, previously mentioned, we validated an efficient C++ implementation to determine the ultimate practical computational performance of our approach. On the same simulated lap as before, the C++ SBMP planner was used in MPC fashion and computational performance for all SBMP planner calls is presented in Fig. 17. These are the results achieved on the computer with Intel i5

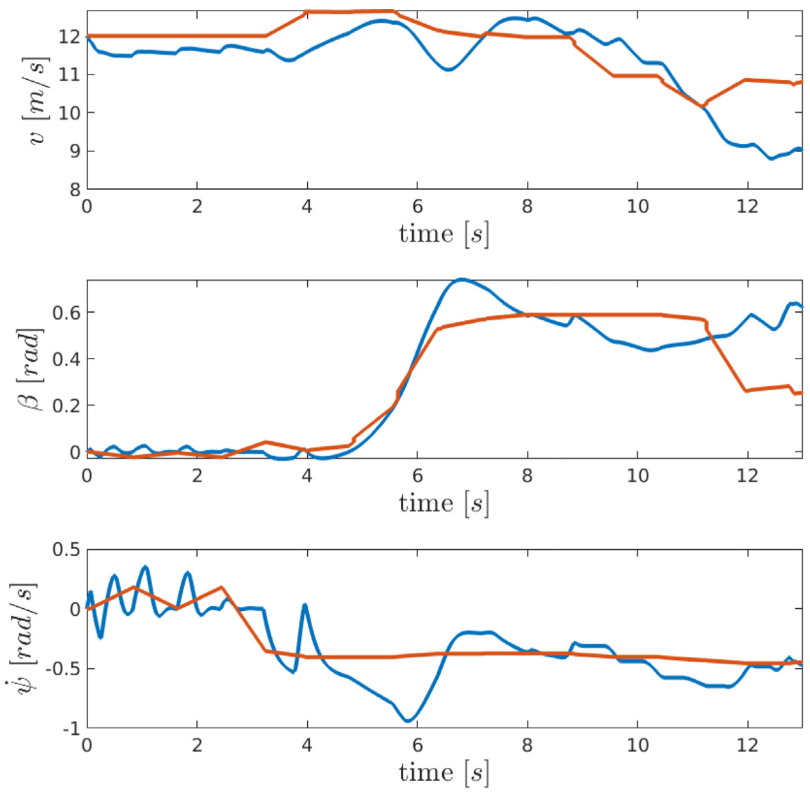
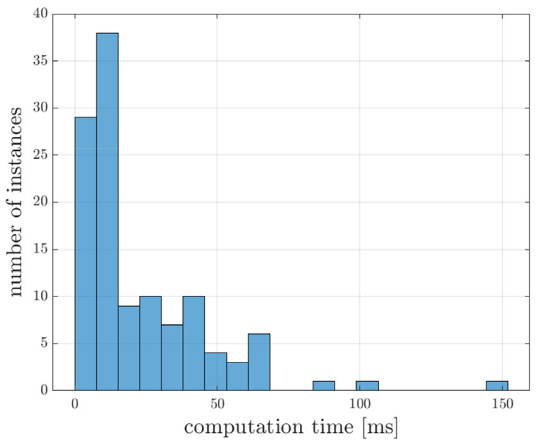


Fig. 16. Tracking reference trajectory using path tracking and drift controller (state trajectory).



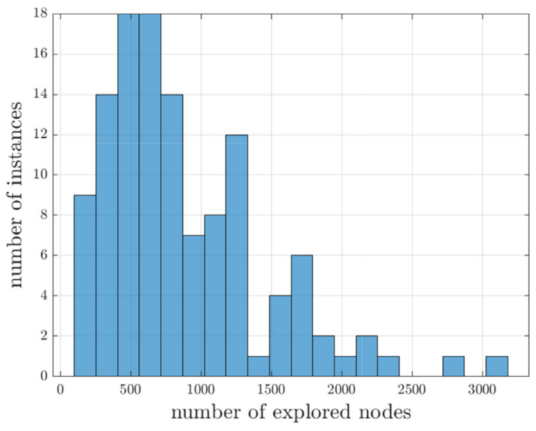


Fig. 17. Histogram of computation times (left) and explored nodes (right) for the track.

8th generation CPU, with 8 Gb RAM. As can be seen in the left figure, all planning computations are under 100 ms with a median time of 13.75 ms and mean time of 22.95 ms. As can be seen in the right figure, all computations explored under 3500 nodes to find the solution, with a median of 716 nodes.

Additionally, we performed an extensive simulation study to analyze the practical computational complexity of the algorithm in terms of horizon length and search space size. As mentioned earlier, there is a trade-off between computational time and solution quality measured in the lap time. Therefore we show both of them in Fig. 18. For analyzing the sensitivity on the horizon length, we preserve well-tuned search-space discretization and vary only the horizon length. To be able to keep the vehicle on the road (i.e. provide a sufficient planning horizon in the future), the time horizon in MPC is fixed. So, as we change the horizon length (number of motion primitives), we also adapt accordingly the time step length of motion primitives, so that their product is constant. Experimental results from 139 laps are shown in Fig. 18 (right). The results indicate that the computation time increases as we increase the horizon, following the exponential trend. There is no clear trend in the lap time and the mean time is rather constant with variations across multiple runs. It is worth noting that for horizons less than 4 steps, the vehicle is not able to drive the full lap without losing control.

For analyzing the sensitivity on search-space size, we vary discretization steps for each of the state variables so we have different sizes of the grid that represent $\mathcal{X}$. Experimental results over 50 laps (each lap 50+ planning instances) are shown in Fig. 18 (left). The results indicate that computational time increases as we increase the horizon, following the linear trend (on a selected range). On the other hand, the lap time marginally improves after cca 700k states. It is important to note that the results are slightly misleading as the search-space size

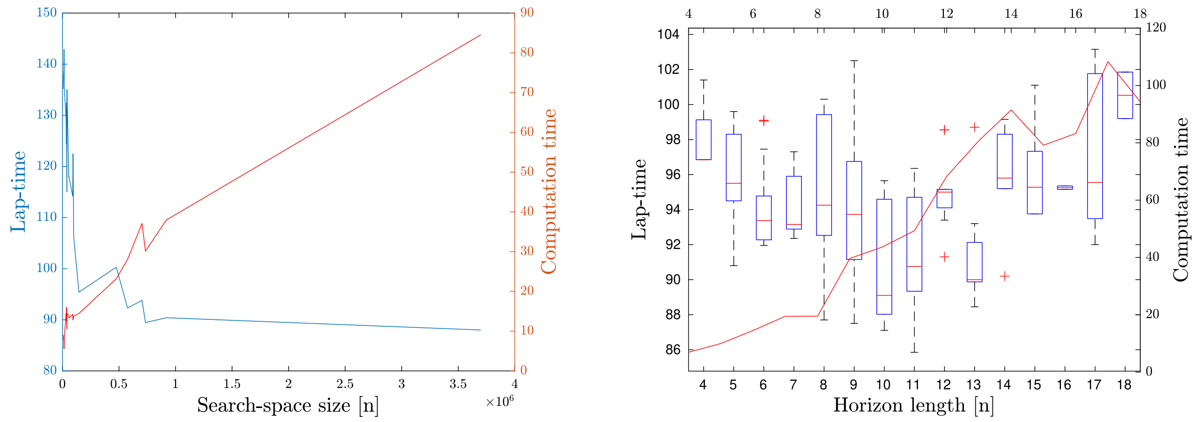


Fig. 18. Computational time and lap time depending on the search-space size (left) and horizon length (right).

actually scales exponentially with the number of discretization steps for state variables.

This analysis provides a deeper insight into the computation complexity of our approach. It is important to highlight again that we show practically that close-to-optimal lap time can be achieved with acceptable computation times for a reasonable planner setting.

5. Conclusions and outlook

In this paper, we presented the SBMP, a novel A* search-based task and motion planning approach that enables agile automated driving on a slippery surface. The proposed method enables us to extend state-of-the-art approaches for drift-like driving from a steady-state drifting on a single curve to continuous driving on the arbitrary road, effectively entering (or exiting) drifting maneuvers and switching between right and left turns. The SBMP consists of tree search, efficient generation of dynamically feasible successor nodes based on motion primitives, and the mode-feasibility map that enables us to use multiple locally approximated models for motion primitives generation. In this way, SBMP treats this problem as TAMP and effectively decides *when* to go into which *mode* and *how* to execute it. The proposed method assumes that the vehicle parameters and the road surface properties are known to a certain degree, which allows to define a set of steady-state cornering maneuvers. The method is evaluated on a mixed circuit characterized by slippery conditions (gravel), which contains several road sections of varying curvature radii R_c. In several instances, due to the particular road surface considered, the optimal selected trajectory involves drifting, which in certain conditions ensures the maximum lateral acceleration. Such results demonstrate the capability of the proposed SBMP to generate feasible close-to-optimal trajectories on slippery conditions while considering a limited prediction horizon. Moreover, when considering U-turns with curvature radius as tight as 15 m, trajectories are comparable in shape to the ones obtained by e.g., Tavernini et al. (2013), when the full segment is optimized offline in order to find the minimum time optimal maneuver.

Future research direction might utilize other ways to learn the Equilibrium State Manifold e.g., for real vehicles from human experts — Learning from Demonstration (LfD) or learning rapid generation of local car maneuvers similar to Kicki et al. (2021), and validated on a real vehicle similar to Ajanovic et al. (2020). The sub-optimal policy could be further improved by learning from experience e.g., improving our base controller (Goh et al., 2019) with Residual Policy Learning (Silver et al., 2019), a fully RL-based controller (Cai et al., 2020) or learning from corrections in an Interactive Imitation Learning fashion (Celemin et al., 2022). This might help to improve lap-time performance and to generalize to the distribution shift (e.g., changing tire–road conditions). Policy execution could be further improved by making a tighter connection between controllers and motion planning, e.g., by considering delays of controllers in the planning stage. Furthermore, the robustness might be improved by considering non-deterministic models. Future research directions might also consider more challenging scenarios such as multiple vehicles on the road in a race, where besides dynamics game-theoretic aspect should be considered in a minimax fashion. Our approach is well suited for non-deterministic and game theoretic extensions as it relies on the tree search. Considering them might increase the computational complexity of the problem, but more advanced search algorithms or learning of heuristic functions similar to Ajanović et al. (2019) might help with that. Finally, as this approach is general, it would be useful to see its applicability to other agile robotic problems with similar structures and the extension of this approach with symbolic variables.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The project leading to this study has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675999, ITEAM project. VIRTUAL VEHICLE Research Center is funded within the COMET - Competence Centers for Excellent Technologies - programme by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET programme is administrated by FFG. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. We would like to thank Reviewers for taking the time and effort necessary to review the manuscript. We sincerely appreciate all valuable comments and suggestions, which helped us to improve the quality of the manuscript. Also, we would like to thank Rodrigo Péerez-Dattari and Pablo Borja for their comments on this work.

References

Acosta, M., Ivanov, V., Malygin, S., 2019. On highly-skilled autonomous competition vehicles: An FSM for autonomous rallycross. In: 2019 IEEE International Conference on Mechatronics (ICM), Vol. 1. IEEE, pp. 556–561.

Adabala, B., Ajanovic, Z., 2020. A multi-heuristic search-based motion planning for autonomous parking. In: 30th International Conference on Automated Planning and Scheduling: Planning and Robotics Workshop.

Ajanović, Z., Beglerovic, H., Lacevic, B., 2019. A novel approach to model exploration for value function learning. In: RSS 2019 Workshop on Combining Learning and Reasoning – Towards Human-Level Robot Intelligence. ArXiv Preprint arXiv:1906.02789.

Ajanovic, Z., Klomp, M., Lacevic, B., Shyrokau, B., Pretto, P., Islam, H., Stettinger, G., Horn, M., 2020. Validating SuperHuman automated driving performance. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 3860–3867. http://dx.doi.org/10.1109/SMC42975.2020.9282822.

Ajanovic, Z., Lacevic, B., Shyrokau, B., Stolz, M., Horn, M., 2018. Search-based optimal motion planning for automated driving. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 4523–4530.

Ajanovic, Z., Regolin, E., Stettinger, G., Horn, M., Ferrara, A., 2019. Search-based motion planning for performance autonomous driving. In: The IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks. Springer, pp. 1144–1154.

Bellman, R., Dreyfus, S., 1962. Applied Dynamic Programming. Princeton:[sn].

Betz, J., Wischnewski, A., Heilmeier, A., Nobis, F., Stahl, T., Hermansdorfer, L., Lohmann, B., Lienkamp, M., 2019. What can we learn from autonomous level-5 motorsport? In: Pfeffer, P. (Ed.), 9th International Munich Chassis Symposium 2018. In: Proceedings, Springer Vieweg, Wiesbaden, pp. 123–146.

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., Mangharam, R., 2022. Autonomous vehicles on the edge: A survey on autonomous vehicle racing, 3, 458–488 http://dx.doi.org/10.1109/OJITS.2022.3181510.

Cai, P., Mei, X., Tai, L., Sun, Y., Liu, M., 2020. High-speed autonomous drifting with deep reinforcement learning, 5, 1247–1254 http://dx.doi.org/10.1109/LRA.2020.2967299.

Celemin, C., Pérez-Dattari, R., Chisari, E., Franzese, G., de Souza Rosa, L., Prakash, R., Ajanović, Z., Ferraz, M., Valada, A., Kober, J., et al., 2022. Interactive imitation learning in robotics: A survey. Found. Trends® Robot. 10 (1–2), 1–197.

Fikes, R.E., Nilsson, N.J., 1971. STRIPS: A new approach to the application of theorem proving to problem solving, 2, 189–208.

Frazzoli, E., Dahleh, M.A., Feron, E., 2002. Real-time motion planning for agile autonomous vehicles. J. Guid. Control Dyn. (ISSN: 0731-5090) 25 (1), 116–129.

Garrett, C.R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L.P., Lozano-Pérez, T., 2021. Integrated task and motion planning. Annu. Rev. Control Robot. Auton. Syst. 4, 265–293.

Genta, G., 1997. Motor Vehicle Dynamics: Modeling and Simulation, Vol. 43. World Scientific.

Goh, J.Y., Goel, T., Christian Gerdes, J., 2019. Toward automated vehicle control beyond the stability limits: Drifting along a general path. J. Dyn. Syst. Meas. Control (ISSN: 0022-0434) 142 (2).

Hart, P., Nilsson, N., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), http://dx.doi.org/10.1109/TSSC.1968.300136.

Kicki, P., Gawron, T., Ćwian, K., Ozay, M., Skrzypczyński, P., 2021. Learning from experience for rapid generation of local car maneuvers. Eng. Appl. Artif. Intell. 105, 104399.

Kolter, J.Z., Plagemann, C., Jackson, D.T., Ng, A.Y., Thrun, S., 2010. A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. In: 2010 IEEE International Conference on Robotics and Automation. IEEE, pp. 839–845.

Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., How, J., 2008. Motion planning in complex environments using closed-loop prediction. In: AIAA Guidance, Navigation and Control Conference and Exhibit. p. 7166.

Li, B., Ouyang, Y., Li, L., Zhang, Y., 2022. Autonomous driving on curvy roads without reliance on frenet frame: A cartesian-based trajectory planning method. IEEE Trans. Intell. Transp. Syst..

Liniger, A., Domahidi, A., Morari, M., 2015. Optimization-based autonomous racing of 1:43 scale RC cars. Optim. Control Appl. Methods 36 (5), 628–647. http://dx.doi.org/10.1002/oca.2123.

Liniger, A., Lygeros, J., 2019. A noncooperative game approach to autonomous racing. IEEE Trans. Control Syst. Technol. 1–14. http://dx.doi.org/10.1109/TCST.2019.2895282.

Liu, S., Mohta, K., Atanasov, N., Kumar, V., 2018. Search-based motion planning for aggressive flight in se (3). IEEE Robot. Autom. Lett. 3 (3), 2439–2446.

Lu, Z., Shyrokau, B., Boulkroune, B., van Aalst, S., Happee, R., 2018. Performance benchmark of state-of-the-art lateral path-following controllers. In: 2018 IEEE 15th International Workshop on Advanced Motion Control (AMC). pp. 541–546. http://dx.doi.org/10.1109/AMC.2019.8371151.

Mandalika, A., Salzman, O., Srinivasa, S., 2018. Lazy receding horizon A* for efficient path planning in graphs with expensive-to-evaluate edges. In: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 28. pp. 476–484.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. PDDL-the planning domain definition language.

Montemerlo et al., M., 2008. Junior: The stanford entry in the urban challenge. J. Field Robotics 25 (9), 569–597. http://dx.doi.org/10.1002/rob.20258.

Pacejka, H., 2012. Tire and Vehicle Dynamics. Butterworth-Heinemann.

Perumal, P.S., Sujasree, M., Chavhan, S., Gupta, D., Mukthineni, V., Shimgekar, S.R., Khanna, A., Fortino, G., 2021. An insight into crash avoidance and overtaking advice systems for autonomous vehicles: A review, challenges and solutions. Eng. Appl. Artif. Intell. 104, 104406.

Ranganeni, V., Chintalapudi, S., Salzman, O., Likhachev, M., 2020. Effective footstep planning using homotopy-class guidance. Artificial Intelligence 286, 103346.

Regolin, E., Vazquez, A.G.A., Zambelli, M., Victorino, A., Charara, A., Ferrara, A., 2019. A sliding mode virtual sensor for wheel forces estimation with accuracy enhancement via EKF. IEEE Trans. Veh. Technol. 68 (4), 3457–3471.

Regolin, E., Zambelli, M., Ferrara, A., 2018. A multi-rate ISM approach for robust vehicle stability control during cornering. IFAC-PapersOnLine 51 (9), 249–254.

Russell, S., Norvig, P., 2021. Artificial Intelligence: A Modern Approach, fourth ed..

Sharma, O., Sahoo, N.C., Puhan, N., 2021. Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. Eng. Appl. Artif. Intell. 101, 104211.

Silver, T., Allen, K., Tenenbaum, J., Kaelbling, L., 2019. Residual policy learning. http://dx.doi.org/10.48550/arXiv.1812.06298, arXiv:1812.06298.

Sutton, R.S., Precup, D., Singh, S., 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning, 112, 181–211. http://dx.doi.org/10.1016/S0004-3702(99)00052-1.

Tavernini, D., Massaro, M., Velenis, E., Katzourakis, D.I., Lot, R., 2013. Minimum time cornering: the effect of road surface and car transmission layout. Veh. Syst. Dyn. 51 (10), 1533–1547.

Valls, M.I., Hendrikx, H.F., Reijgwart, V.J., Meier, F.V., Sa, I., Dube, R., Gawel, A., Burki, M., Siegwart, R., 2018. Design of an autonomous racecar: Perception, state estimation and system integration. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 2048–2055. http://dx.doi.org/10.1109/ICRA.2018.8462829.

Velenis, E., Katzourakis, D., Frazzoli, E., Tsiotras, P., Happee, R., 2011. Steady-state drifting stabilization of RWD vehicles. Control Eng. Pract. 19 (11), 1363–1376.

Velenis, E., Tsiotras, P., Lu, J., 2007. Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn. In: ECC. IEEE, pp. 1233–1240.

Velenis, E., Tsiotras, P., Lu, J., 2008. Optimality properties and driver input parameterization for trail-braking cornering. Eur. J. Control 14 (4), 308–320. http://dx.doi.org/10.3166/ejc.14.308-320.

Werling, M., Kammel, S., Ziegler, J., Gröll, L., 2012. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. Int. J. Robot. Res. 31 (3), 346–359. http://dx.doi.org/10.1177/0278364911423042.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J.M., Boots, B., Theodorou, E.A., 2017. Information theoretic MPC for model-based reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 1714–1721.

You, C., Tsiotras, P., 2018. Real-time trail-braking maneuver generation for off-road vehicle racing. In: 2018 Annual American Control Conference (ACC). IEEE, pp. 4751–4756.

Youakim, D., Cieslak, P., Dornbush, A., Palomer, A., Ridao, P., Likhachev, M., 2020. Multirepresentation, Multiheuristic A* search-based motion planning for a free-floating underwater vehicle-manipulator system in unknown environment. J. Field Robotics 37 (6), 925–950.

Zhang, F., Gonzales, J., Li, S.E., Borrelli, F., Li, K., 2018. Drift control for cornering maneuver of autonomous vehicles. Mechatronics 54, 167–174.

Ziegler, J., Bender, P., Dang, T., Stiller, C., 2014. Trajectory planning for Bertha — A local, continuous method. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings. IEEE, ISBN: 978-1-4799-3638-0, pp. 450–457.

Ziegler, J., Stiller, C., 2010. Fast collision checking for intelligent vehicle motion planning. In: Intelligent Vehicles Symposium (IV), 2010 IEEE. IEEE, pp. 518–522.