Time-Dependent Hybrid-State A* and Optimal Control for Autonomous Vehicles in Arbitrary and Dynamic Environments [⋆]

Andreas Folkers [*] **Matthias Rick** [*] **Christof Büskens** [*]

[*] *Center for Industrial Mathematics, University of Bremen, Germany, (e-mail: {afolkers, mrick, bueskens}@uni-bremen.de)*

Abstract: The development of driving functions for autonomous vehicles in urban environments is still a challenging task. In comparison with driving on motorways, a wide variety of moving road users, such as pedestrians or cyclists, but also the strongly varying and sometimes very narrow road layout pose special challenges. The ability to make fast decisions about exact maneuvers and to execute them by applying sophisticated control commands is one of the key requirements for autonomous vehicles in such situations. In this context we present an algorithmic concept of three correlated methods. Its basis is a novel technique for the automated generation of a free-space polygon, providing a generic representation of the currently drivable area. We then develop a time-dependent hybrid-state A* algorithm as a model-based planner for the efficient and precise computation of possible driving maneuvers in arbitrary dynamic environments. While on the one hand its results can be used as a basis for making short-term decisions, we also show their applicability as an initial guess for a subsequent trajectory optimization in order to compute applicable control signals. Finally, we provide numerical results for a variety of simulated situations demonstrating the efficiency and robustness of the proposed methods.

Keywords: Autonomous vehicles, Automated guided vehicles, Path planning, Automotive control, Optimal control, Nonlinear model predictive control, Vehicle models, Moving objects

1. INTRODUCTION

The vision of driverless cars comes hand in hand with promises of increased road safety or cost efficiency (Maurer et al., 2016). While the development of such technologies lasts until today, it already started with early pioneering demonstrations in the 1990s (Dickmanns, 1997). However, it is still a challenging task to develop a fully autonomous system that is able to safely drive in very general and dynamic environments, such as unrestricted urban areas with other traffic participants. A robust navigation in such situations relies on the one hand on fast planning of possible actions and on the other hand on the computation of corresponding vehicle controls that lead to an efficient and safe but also comfortable maneuver.

Some approaches for the implementation of such strategies introduce highly specialized solutions for specific maneuvers like parking (e.g. Siedentop et al., 2015). More general solutions might be expected by using data-driven concepts. Most recently, new methods based on deep learning proposed control policies for tasks like lane following (Bojarski et al., 2016) or even more generic navigation (Folkers et al., 2019). Although these algorithms show promising results, their application in unknown situations strongly depends on the diversity of the data in the training procedure.

A very general approach to solve arbitrary navigation tasks is the usage of optimal control techniques for model-based planning. This method proved to provide very sophisticated solutions in various context such as the control of spacecrafts (e.g. Schattel, 2018) or mobile robots (e.g. Meerpohl et al., 2019). Most of all it emerged to be well applicable for the real-time control of autonomous vehicles in the setting of nonlinear model predictive controllers (e.g Falcone et al., 2007; Paden et al., 2016; Rick et al., 2019).

Certainly, since optimal control methods are usually based on local optimization, their efficient application relies heavily on a well thought out implementation of the corresponding problem. For autonomous driving this is strongly related to the description of spatial constraints given by e.g. static and moving obstacles or the boundary of the lane. In well structured scenarios, which is for example the case in highway driving, methods for obstacle avoidance can be based on geometric considerations (e.g. Wolf and Burdick, 2008). However, for more general cases Meerpohl et al. (2019) propose a description of the environment based on free-space polygons. This, once again, has been proven to be well suited for solving optimal control problems in the context of autonomous vehicles (Sommer et al., 2018). Nevertheless, it should be noted that all solutions proposed so far must be provided with a set of viewpoints from which the polygon is then created, which can be a disadvantage in very general applications.

A further crucial element in solving optimal control problems is the requirement of a sufficiently good initial estimate of the solution (Shiller, 2015). Ideally, this would take into account both the observed spatial restrictions as well as the constraints on the vehicle's motion, defined by the considered dynamic model. The latter condition might be regarded by analytically computing a shortest path according to Reeds and Shepp (1990)

[⋆] The authors would like to thank the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) and the German Aerospace Center (DLR) Space Administration for supporting this work (grant no. 50 NA 1909).

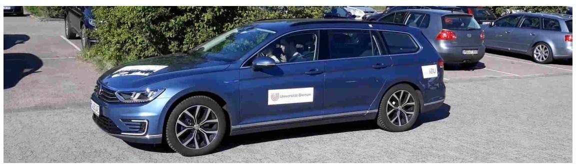


Fig. 1. The research vehicle.

as done by Sommer et al. (2018). However, this does not incorporate any obstacles which might affect the convergence of a local optimizer. Alternatively, one could utilize a fast planning algorithm like A* for finding a reference path in a static environment. This could be even further enhanced by using, e. g., the hybrid-state A* method introduced by Dolgov et al. (2008), which also takes a dynamic model of the vehicle into account. It is worth pointing out that, even doing so, this initial guess would still not be able to capture the movement of dynamic traffic participants.

1.1 Paper Subjects

The contribution of this work is threefold. On the one hand, we will develop an extension of the hybrid-state A* algorithm to allow fast path planning and decision making in dynamic environments. On the other hand, we show how to use its results as a reference for a consecutive trajectory optimization step which is, in turn, the basis for a model predictive controller. The foundation of both, planning and optimization, lies in the description of the vehicles (arbitrary) static surrounding by a free-space polygon. Therefore, our third contribution is the introduction of a method for its automated generation, which makes this technique applicable as a preliminary step without further knowledge.

1.2 Research Vehicle

The methods presented in this paper represent a general approach to motion planning and the computation of control commands for self-driving cars in urban environments. However, it should be mentioned that specific vehicle parameters used for the evaluation - like control constraints, signal delays, sizes, etc. - refer to our research vehicle shown in Fig. 1. It allows the execution of autonomous driving maneuvers, for example, by providing corresponding acceleration values and steering angles. See Rick et al. (2019) for more details.

2. AUTOMATED FREE-SPACE POLYGON GENERATION

In this section, we describe how a free-space polygon can be generated for the static components of an arbitrary environment. Even if we take the example of autonomous driving in this work, we would like to emphasize that this method is applicable to general planning problems in the plane. Thereby, perception might be based on both, measured sensor data as well as a-priori knowledge about fixed parts of the current surrounding (e. g. the position of lanes). The union of these informations will in the following be denoted as $\mathcal{D}$ and is assumed to be a point cloud to facilitate general applicability.

The algorithm itself can be understood as an iterative process, that increases the size of the free-space polygon in every step. At each stage the polygon is expanded according to selected viewpoints. Afterwards new viewpoints are identified based on the latest expansion. Compared to, e. g., flood filling algorithms,

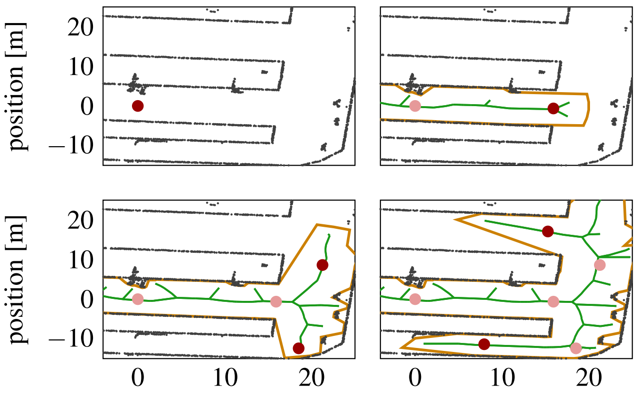


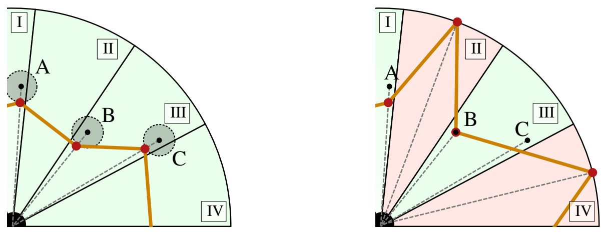
Fig. 2. Automated free-space polygon generation algorithm on real data from a parking lot (black). Orange lines show the first three polygons from top left to bottom right. Red dots illustrate the computed viewpoints to be used for the next iteration. Light red dots show previously used viewpoints. Green lines correspond to the respective Voronoi paths $\mathcal{R}$.

this approach focuses on the generation of the local free-space boundary only and is therefore very efficient and due to its iterative character easily adjustable in search direction or depth.

The method is initialized with the systems's current state s_0 (position and orientation) being the starting viewpoint. An illustration of the first three steps of an exemplary situation of the whole procedure is given in Fig. 2. The complete method is presented in Algorithm 1.

2.1 Generation of Polygons

For each given viewpoint υ a free-space polygon is created and afterwards united with the polygons from previous steps of the automated generation process. The creation of a single polygon is based on the entries of the obstacle point cloud in the vicinity of the corresponding viewpoint and can be implemented by various methods, e. g. the one intro duced by Meerpohl et al. (2019). Therein, the vertices of a free-space polygon are defined by emitting circles on rays and testing them for collisions with obstacle points. However, depending on the number of point cloud entries, other approaches may be more efficient. For example, the amount of necessary evaluations could be reduced by assigning the obstacle points to circular sectors. In this paper we apply such a *circular sector*-approach which also



(a) Considering the obstacle size. (b) Disregarding the obstacle size.

Fig. 3. Free-Space Polygon Generation. Grey circles represent obstacles points with their specific size and the black dots mark their centers. The dashed lines show the direct connection between obstacle point and viewpoint. Considered obstacles are marked by red dots. Empty circular sectors are colored red, otherwise green. The resulting free-space polygon is given by the orange line.

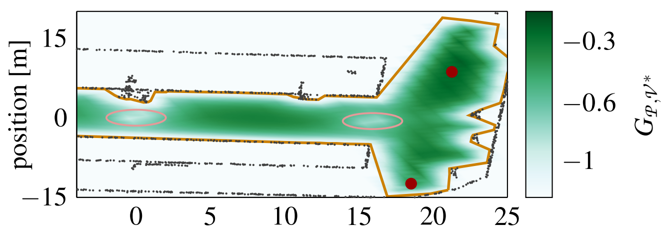


Fig. 4. Potential function $G_{\mathcal{P},\mathcal{V}^*}$ for finding new viewpoints (represented as red dots) after the second stage of the polygon generation shown in Fig. 2. The light red ellipses indicate the position of previous viewpoints.

considers the obstacle point sizes. For this purpose, the position of the nearest point cloud entry in each sector is reduced to its closest point to the viewpoint, as illustrated in Fig. 3a. In case of no occupancy, a point is selected according to the maximal predefined expansion ξ. However, it is possible that an obstacle point extends across more than one sector. In this case, it is considered in each of them and hence might be marked more than once. Because of this reason, for instance, obstacle point B is assigned to sectors II and III and obstacle point C to sector IV in Fig. 3a. If obstacle sizes were not taken into account like this, it could happen that a sector would be classified as free, leading to undesired results, as shown in Fig. 3b.

2.2 Generation of Viewpoints

The selection of valuable new viewpoints has to regard the following two conflicting criteria. On the one hand, they should be close to the boundary of the current-stage polygon to promote further expansion. However, on the other hand, if a viewpoint is too close to the boundary, its expansion might be strongly hindered by obstacles directly next to it.

We propose the following procedure to find a good trade-off between both aspects. First, local maximizers of the potential function $G_{\mathcal{P},\mathcal{V}^*} : \mathbb{R}^2 \to [-\infty, 0)$ defined by

$$G_{\mathcal{P},\mathcal{V}^*}(p) = \begin{cases} -\frac{1}{1+\mathrm{d}_{\mathcal{P}}(p)} - \sum\limits_{\upsilon \in \mathcal{V}^*} \frac{1}{1+\mathrm{d}(\upsilon,p)}, & \text{if } p \text{ is inside } \mathcal{P} \\ -\infty, & \text{else} \end{cases}$$

are considered as candidates. Here, $\mathrm{d}(\cdot,\cdot)$ and $\mathrm{d}_{\mathcal{P}}(\cdot)$ define the distances of a point to another one or to the polygon's border, respectively. $\mathcal{V}^*$ denotes the set of previous viewpoints. The benefit of this potential is that its maximizers are placed in the center of areas that are far away from old reference points.

However, it is still possible that there are local maximizers of $G_{\mathcal{P},\mathcal{V}^*}$ that are rather close to each other or, alternatively, between old viewpoints. In artificially created examples without a road-like structure, solutions might even be represented by line segments. Therefore, to avoid accumulations of viewpoints, a subset of the new candidates is selected that complies with a predefined clearance γ in a second step. This selection may, for instance, be based on the corresponding values of $G_{\mathcal{P},\mathcal{V}^*}$, a given target route or the current direction of travel. The result is illustrated for an exemplary situation in Fig. 4 which shows the potential field of the polygon from Fig. 2 (bottom left). Here two new viewpoints are determined.

3. TIME-DEPENDENT HYBRID-STATE A^*

An essential feature of self-driving cars is the ability to rapidly plan a set of executable actions even in complex situations. A

Algorithm 1: Automated free-space polygon generation.

```
Input: sytem state $s_0$, data $\mathcal{D}$
Parameter: refinements $\tau$, clearance $\gamma$, expansion $\xi$

Set polygon $\mathcal{P} := \{\}$ and viewpoints $\mathcal{V} := \{s_0\}, \mathcal{V}^* := \{\}$;
for $k \leftarrow 0$ to $\tau$ do
    Set $i := 0$;
    foreach $\upsilon \in \mathcal{V}$ do
        $i \leftarrow i + 1$;
        $\mathcal{P}_i :=$ GeneratePolygonFrom($\upsilon, \mathcal{D}$);
    $\mathcal{P} \leftarrow$ unite($\mathcal{P}, \mathcal{P}_1, \ldots, \mathcal{P}_i$);
    if $k \neq \tau$ then
        $\mathcal{V}^* \leftarrow \mathcal{V}^* \bigcup \mathcal{V}$;
        $\mathcal{V} \leftarrow$ GetViewpoints($\mathcal{P}, \mathcal{V}^*$);
return $\mathcal{P}$;
Function GeneratePolygonFrom($\upsilon, \mathcal{D}$)
    $\mathcal{P} = \{\}$;
    foreach $S \in$ Sectors do
        Set $p_{\text{nearest}} :=$ nan, $d_{\text{nearest}} := \infty$;
        foreach $p \in \mathcal{D}$ do
            if $p \in S$ and $\mathrm{d}(p,\upsilon) < d_{nearest}$ then
                $p_{\text{nearest}} \leftarrow p$, $d_{\text{nearest}} \leftarrow \mathrm{d}(p,\upsilon)$;
        if $p_{nearest} \neq nan$ then
            Add $p_{\text{nearest}}$ to $\mathcal{P}$;
        else
            Add midpoint at end of sector to $\mathcal{P}$;
    return $\mathcal{P}$;
Function GetViewpoints($\mathcal{P}, \mathcal{V}^*$)
    Set $\mathcal{V} \leftarrow \{p \in \mathbb{R}^2 \mid p$ local maximizer of $G_{\mathcal{P},\mathcal{V}^*}\}$;
    while $M := \{p \in \mathcal{V} \mid \mathrm{d}(p, \mathcal{V}^* \bigcup [\mathcal{V} \setminus \{p\}]) < \gamma\} \neq \emptyset$ do
        $\mathcal{V} \leftarrow \mathcal{V} \setminus \{p\}$ for one $p \in M$ ;  // no accumulations
    return $\mathcal{V}$;
```

well-known approach that fulfills these requirements in static settings is the hybrid-state A* algorithm by Dolgov et al. (2008). Building upon its idea of hybrid nodes, we present details of an extension that is able to also capture time-varying parts of the vehicles environment.

3.1 Kinematics

To consider the movement of objects O in the planned path of the ego-vehicle, it is necessary to take its speed into account and to allow changes to it. The simplest kinematic description incorporating this is the single-track model (e.g. Luca et al., 1998) given as

$$\left(\dot{x}, \dot{y}, \dot{\phi}, \dot{v}\right)^\top = \left(v\cos(\phi), v\sin(\phi), {}^{v}\!/\!{}_{L}\tan(\beta), a\right)^\top. \quad (1)$$

Therein, the car's state consists of the (x,y)-position (defined as the center of the rear axle), the orientation ϕ as well as the speed v. The wheelbase L denotes a vehicle-specific parameter. The motion of the car may be influenced by the acceleration a or by the steering angle β.

Like in the ordinary hybrid-state A* algorithm, the motion of the car is regarded by applying discrete controls, in our case $\mathcal{A} := \{a_1, \ldots, a_{N_a}\}$ and $\mathcal{B} := \{\beta_1, \ldots, \beta_{N_\beta}\}$. This results in a continuous state representation (t, x, y, ϕ, v), with t denoting the time (in order to take varying locations of obstacles into account). The A* search space is then formed by discretization of these state variables, whereby the continuous representation and its corresponding discrete cell are stored jointly. After

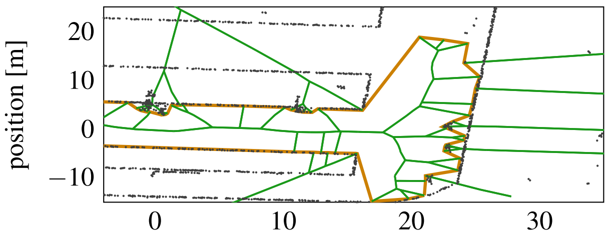


Fig. 5. (Unfiltered) Voronoi diagram (green lines) for the second free-space polygon (orange lines) from Fig. 2.

termination of the search algorithm, a feasible path in the sense of the kinematic model (1) is available by following the continuous representation of the corresponding nodes.

3.2 Dynamic Voronoi Field

As proposed in the original hybrid-state A* method, our approach is guided by a cost function based on a Voronoi diagram. During planning, its edges serve as a reference path to avoid collisions. Moreover, a Voronoi diagram for a free-space polygon can be computed very efficiently by the Sweepline algorithm (e.g. Sydorchuk, 2012), which allows the input sites to be solely in the form of segments. The resulting set of edges, which come typically in the form of curves, are then approximated by straight lines to enable the fast computation of distances to them.

The resulting Voronoi diagram usually contains edges that are not suitable as references for path planning., e.g., those that lie outside or close to the border of the polygon, as shown in Fig. 5. However, the corresponding vertices can be eliminated easily, so that the Voronoi reference paths $\mathcal{R}$ presented in Fig. 2 remain. Note that the small branch-offs shown could be filtered out further, but have almost no influence on the planning result in practice, as they usually do not comply with the vehicles kinematics (1).

In cases where other dynamic objects O are *within* the current free-space polygon (at least partially), Voronoi reference paths $\mathcal{R}_t$ are computed for a number of discrete times to take their movement into account as well. This is easy to parallelize and can therefore be conducted very efficiently. The result is then used to define the time-dependent Voronoi field

$$\rho_{\mathcal{R}_t,\mathcal{P},O_t}(s_t) := \left(\frac{\alpha}{\alpha + \mathrm{d}^3_{\mathcal{P},O_t}(s_t)}\right)\left(\frac{\mathrm{d}^2_{\mathcal{R}_t}(s_t)}{\mathrm{d}^3_{\mathcal{P},O_t}(s_t) + \mathrm{d}^2_{\mathcal{R}_t}(s_t)}\right) \left(\frac{\mathrm{d}^3_{\mathcal{P},O_t}(s_t) - \mathrm{d}_{\max}}{\mathrm{d}_{\max}}\right)^2 \tag{2}$$

for the vehicle state s_t at the discrete time t and with $\alpha, \mathrm{d}_{\max} \in \mathbb{R}_{>0}$. Further, $\mathrm{d}^2_{\mathcal{R}_t}(\cdot)$ denotes the distance from the vehicle's reference point (x_t, y_t) to the Voronoi path. Moreover, $\mathrm{d}^3_{\mathcal{P},O_t}(\cdot)$ represents the minimal distance of the ego-vehicle to the polygon $\mathcal{P}$ and all dynamic obstacles O_t, dependent on (x_t, y_t, ϕ_t).

The definition in (2) has similar characteristics as the potential introduced by Dolgov et al. (2008), i.e. $\rho_{\mathcal{R}_t,\mathcal{P},O_t} \in [0,1]$ and its property to support the navigation in narrow areas. However, it is a generalization in the sense that it considers time-varying information and performs full collision checks. An example of a maneuver based on this is given in Fig. 6.

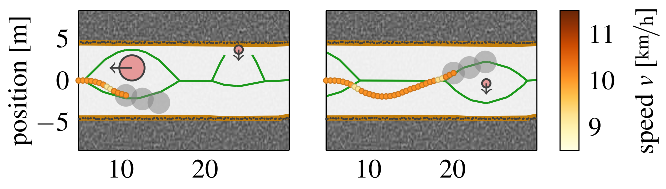


Fig. 6. Avoiding moving obstacles (represented as pale red circles) using the dynamic Voronoi path (green lines). The planning result is shown by the sequences of colored dots at two different times. The orange lines show the free-space polygon and the three gray circles are used to check for collisions of the ego-vehicle.

3.3 Costs and Heuristics

Given a desired goal state s_T, the movement cost g to a node s_t in our proposed A* search penalizes three things:

(i) Deviations from a desired speed v_{set}:
$$g_v := \frac{(v_t - v_{\text{set}})^2}{\max(v^2_{\text{set}}, v^2_{\text{set, min}})}.$$
(ii) Discounted proximity to obstacles:
$$g_o := \rho_{\mathcal{R}_t,\mathcal{P},O_t}(s_t)\frac{p_{\circledcirc}}{p_{\oplus}}.$$
(iii) Making a step:
$$g_p := 1.$$

The lower limit $v_{\text{set, min}}$ for the reference speed is introduced to avoid singularities at small values (e.g. when stopping the car). The Voronoi field $\rho_{\mathcal{R}_t,\mathcal{P},O_t}$ is discounted based on the expected remaining relative path length. This is obtained as the ratio of $p_{\circledcirc}$ and $p_{\oplus}$, which represent estimates of the absolute path lengths to the goal node s_T from the current state s_t and from the initial node, respectively. To approximate these two quantities, we employ the obstacle-free shortest path proposed by Dubins (1957) (for the case of driving forward) or Reeds and Shepp (1990) (if driving backwards is also allowed, e.g. during parking). Finally, all three components are normalized with the relative stepsize $p_{\ominus}/p_{\oplus}$, which can be assumed fixed for a constant discretization of the A* search space. Altogether, the costs g are then defined as the weighted sum

$$g := (w_v g_v + w_o g_o + w_p g_p)\frac{p_{\ominus}}{p_{\oplus}}, \quad w_v, w_o, w_p \in \mathbb{R}_{\geq 0}.$$

The A* heuristic h, on the other hand, should estimate the total costs to reach the goal s_T from a given node. A very optimistic and certainly admissible lower bound for this would be the expected remaining (obstacle-free) path length $p_{\circledcirc}/p_{\oplus}$ for the assumption of having no further deviations from both the reference speed and the Voronoi path. Nevertheless, we propose a more pessimistic approach by expecting the current deviations to persist, hence resulting in the heuristic

$$h := (w_v g_v + w_o g_o + w_p g_p)\frac{p_{\circledcirc}}{p_{\oplus}}.$$

In practice, using h typically provides good approximations of the true costs and thus results in fast convergence of the planner. However, this A* algorithm is not admissible.

4. OPTIMIZATION & OPTIMAL CONTROL

The preceding sections describe how model-based trajectories can be planned in complex and dynamic environments. However, when it comes to the actual control of an autonomous vehicle, additional constraints and optimality criteria may need to be applied to determine appropriate control signals. For example, it makes sense to consider continuous controls whose rate of change is bounded, to meet the comfort requirements of a passenger. In addition, physical aspects, such as the propagation time of signals, should be considered to achieve high precision in the maneuvers performed.

Model predictive control represent a framework for meeting the above-mentioned demands (e.g. Rick et al., 2019). Therein, optimal control problems of the form

$$\begin{aligned} \min_{z,u,T} \quad & J(z,u,T) \\ \text{s.t. } \dot{z}(t) &= f(z(t),u(t)), \\ z_{\min} &\le z(t) \le z_{\max}, \\ u_{\min} &\le u(t) \le u_{\max}, \\ \Psi(z(0)) &= 0, \\ C(z(t),u(t),t) &\le 0 \text{ for all } t \in [0,T] \end{aligned} \tag{OCP}$$

are solved to find states $z \in \mathcal{C}^1([0,T],\mathbb{R}^{n_z})$, controls $u \in \mathcal{C}^0([0,T],\mathbb{R}^{n_u})$ and the process time $T \in \mathbb{R}_{>0}$, that optimize an objective function J, at high frequency. Furthermore, the system's dynamic f is considered and the optimization variables are subject to box constraints, initial conditions Ψ and path constraints C. The main advantage of this problem formulation lies in its universality, which allows the application for various different driving maneuvers.

In the following, we provide details of an optimal control problem formulation which allows to compute sophisticated trajectories, based on the planning results of the time-dependent hybrid-state A* algorithm, with the goal of controlling a real-world vehicle (see also Sect. 1.2).

4.1 Delayed Single-Track Model

To be applicable to our research car, the dynamic vehicle model f should include the propagation time for applying the steering angle β and the acceleration signal a, which are approximately 120 ms and 500 ms respectively. Furthermore, the representation of the dynamics should be kept simple, to enable fast updates in the context of model predictive control. Therefore, we propose the following single-track model

$$\begin{aligned} \dot{x} &= v\cos(\phi), & \dot{v} &= \tilde{a}, & \dot{\tilde{\beta}} &= {(\beta-\tilde{\beta})}/{\Delta t_\beta}, \\ \dot{y} &= v\sin(\phi), & \dot{\beta} &= \omega_\beta, & \dot{\tilde{a}} &= {(a-\tilde{a})}/{\Delta t_a}, \\ \dot{\phi} &= {v}/{L}\tan(\tilde{\beta}), & \dot{a} &= j, \end{aligned} \tag{3}$$

which extends the kinematics in Sect. 3.1 by introducing a delayed acceleration $\tilde{a}$ and steering angle $\tilde{\beta}$ modeled as PT1-elements. The signal propagation time is then incorporated (indirectly) by means of the time lags Δt_a and Δt_β. Moreover, changes in the control values can be bounded by considering the jerk j and the steering angle velocity ω_β, respectively.

4.2 Objective Function

In order to take safety and comfort aspects into account in the solution of (OCP), the objective function J weights different terms against each other. First, large changes in the acceleration and steering angle are avoided by

$$J_{\text{controls}} := w_0 \int_0^T j(t)\,\mathrm{d}t + w_1 \int_0^T \omega_\beta(t)\,\mathrm{d}t, \quad w_0, w_1 \in \mathbb{R}_{>0}.$$

In addition, deviations from the goal speed v_{set} is penalized by

$$J_{\text{states}} := w_2 \int_0^T (v(t) - v_{\text{set}})\,\mathrm{d}t, \quad w_2 \in \mathbb{R}_{>0}.$$

Finally, the goal position is also incorporated within the objective function instead of considering it as a hard constraint. This enhances the flexibility of the optimization and results in

$$J_{\text{goal}} := w_3 \left((x(T) - x_{\text{goal}})^2 + (y(T) - y_{\text{goal}})^2 \right), \quad w_3 \in \mathbb{R}_{>0}.$$

Altogether, the objective function is defined as

$$J := J_{\text{controls}} + J_{\text{states}} + J_{\text{goal}}.$$

The choice of weighting does not only influence the driving behavior but also has a great impact on the convergence rate and robustness of the optimization.

4.3 Constraints & Initial Guess

The latest estimated state of the vehicle defines the initial condition considered in Ψ of (OCP). Because the final state is already regarded by the objective function, no further conditions for the final state at time T are required.

To prevent the vehicle from colliding with obstacles in terms of the path constraints C, the car is approximated by overlapping circles, see also Fig. 6. In particular, static obstacles are taken into account by enforcing to stay within a free-space polygon. Note that the latter can be reused from the motion planner. Furthermore, in contrast to the results presented by Sommer et al. (2018), moving road users are considered in the path constraints as well.

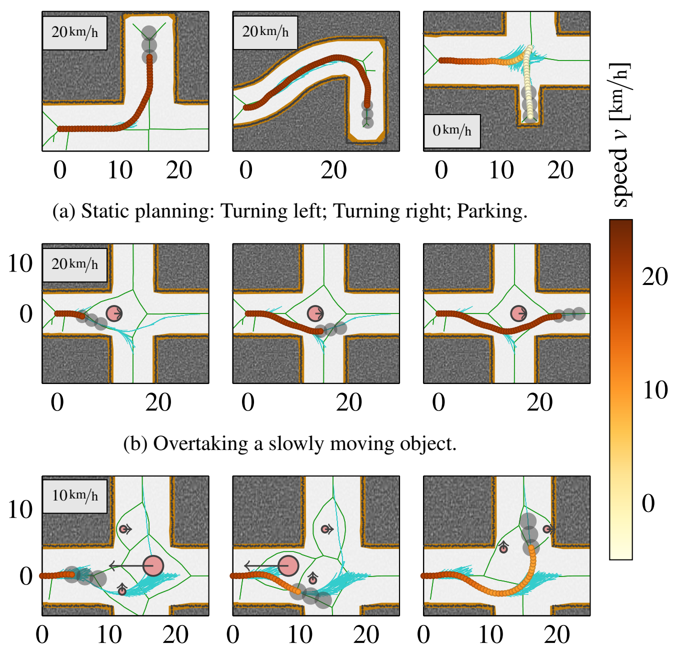
Since the methods for solving optimal control problems usually only find local minima, a good initial guess is required. In the setting of model predictive control, typically the last solution can be used for this. However, if there is a strong change in the goal state provided by the decision making, a complete reoptimization might be necessary. In this case there are several ways to obtain an initial estimate, e.g. by linear interpolation, computing a Reeds-Shepp path or by using the trajectory provided by the time-dependent hybrid-state A* algorithm. We compare the last two options in Sect. 5.2.

5. NUMERICAL RESULTS

We provide numerical results for model-based planning and optimal control in simulated situations, which should exemplarily cover a wide range of requirements on the solver. Thereby, the environment will be represented as a noisy point cloud. Moving obstacles are assumed to be approximated by circles. We restrict their motion to be linear in all cases to allow comparability between different situations. However, all methods are applicable to arbitrary estimates of their movement. The experiments are conducted on a Intel i7-4790 with 3.6 GHz. A summary of all hyperparameters used can be found in Appendix A.

5.1 Path Planning

Given a certain situation, the motion planner performs two preliminary steps: it generates a free-space polygon (see Sect. 2) and precomputes the corresponding Voronoi reference paths for the next 10 s (see Sect. 3.2). A goal is then determined based on the static Voronoi path in accordance with the requirements of a superordinate planner (e.g. *turn left* or *park*). During planning, the costs and heuristics of the time-dependent hybrid-state A*



(a) Static planning: Turning left; Turning right; Parking.

(b) Overtaking a slowly moving object.

(c) Turning and avoiding objects.

Fig. 7. Solutions of the time-dependent hybrid-state A* for exemplary situations. The vehicle's speed is represented by the color gradient of the trajectory. The text boxes show the desired speed v_{set}. Gray circles: ego-vehicle. Red circles: Moving objects. Green lines: Voronoi path. Orange lines: Free-space polygon. Light blue lines: Expanded Nodes. The axis coordinates are in meters.

algorithm are evaluated very efficiently by precalculating the Reeds-Shepp or Dubins paths between discrete states. In addition, the distances to the polygon and reference path are memorized when evaluating the Voronoi potential $\rho_{\mathcal{R}_t,\mathcal{P},O_t}$. On the one hand, this typically reduces the computation time for individual problems by half. On the other hand, these intermediate results can also be used for efficient decision making, which requires multiple evaluations of the same scene with different goal nodes in order to analyze possible actions. Collision checking in the vicinity of obstacles is based on the continuous state representation to reduce discretization errors. Finally, the specific implementation of the A*'s open set combines the advantages of a hashmap and a priority queue, providing fast content checking and access.

Solutions for exemplary static and dynamic scenarios are illustrated in Fig. 7. The corresponding number of expanded nodes and computation times are presented in Table 1. In general, the A* expansion is strongly oriented towards the Voronoi reference path. This leads to very safe maneuvers on the one hand and to a small number of expansions away from the final solution on the other. This also applies to complicated tasks such as parking into a narrow parking space or turning with several moving objects.

The overall computation time results from three components. While the calculation of the free-space polygon is approximately constant for all situations, the creation of the Voronoi path depends on the presence of moving obstacles. If this is the case, dynamic reference paths are generated based on the time discretization. Note that these two auxiliary quantities only have to be determined once for a common scene with different goals. Finally, the execution of the time-dependent hybrid-state

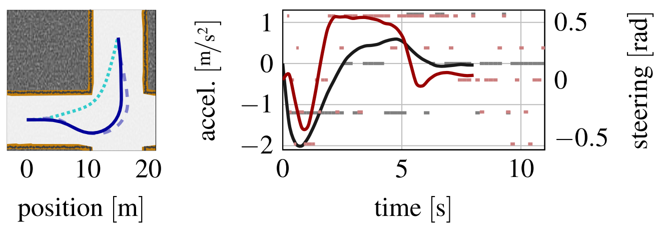


Fig. 8. Solution of (OCP) for the scenario given in Fig. 7c. Left: Resulting optimal path in dark blue. For comparison: The dotted cyan line shows the Reeds-Shepp path and the dashed blue line shows the A* path. Right: Control values - acceleration (black) and steering angle (red). The A* control is represented by the respective pale dashed lines.

A* then terminates for simple situations in less than 5 ms. Its computation time, however, increases with more distant goals, such as when turning right, or when more expansion is required in difficult scenarios. Even in the most demanding cases, solutions are found in less than 40 ms, which makes this method a fast, accurate and robust basis for tactical decisions in urban environments.

5.2 *Optimal Control*

Based on the result of the path planning, the optimal control problem from Sect. 4 is solved to compute executable control signals for a self-driving car. In general, this can be done efficiently by transcribing the formulation (OCP) into a nonlinear optimization problem (NLP) (e.g. Knauer and Büskens, 2019). The resulting task is, in turn, typically characterized by very sparse Jacobian and Hessian matrices which are exploited by highly advanced NLP solvers like WORHP (Büskens and Wassel, 2012).

Fig. 8 illustrates the results of this optimization step for the turning maneuver with obstacles shown in Fig. 7c. For its computation, the existing free-space polygon is reused and WORHP's sequential quadratic programming method (SQP) is applied. One can see, that the optimized turning maneuver deviates from the planned reference solution in its path and being a few seconds shorter. The control values are smooth and especially the steering commands are very targeted if necessary and minimal otherwise.

Although this optimization task is rather complex, the solution shown can be found after 8 SQP steps within 290 ms on the basis of its sophisticated initial estimate. Looking at the same problem without the moving obstacles, as when turning left in

Table 1. Number of expanded nodes and computation times to solve the situations shown in Fig. 7. [·]-brackets: coarser discretization of 1 m for comparison (instead of 0.5 m). †: 3 steps for polygon creation (instead of 2).

		Exp. Nodes		**Computation Time** [ms]		
	Task	**Opened**	**Closed**	**Polygon**	**Voronoi**	**Hybrid A***
Fig. 7a	Turn Left	5106 [4019]	472 [323]	3.0	0.96	4.4 [2.9]
Fig. 7a	Turn Right	22282 [12284]	2618 [1319]	$2.5^{\dagger}$	0.81	20.2 [10.4]
Fig. 7a	Parking	21989	2531	3.1	1.1	38.5
Fig. 7b		3604	335	2.9	15.5	3.5
Fig. 7c		24918	2847	2.9	21.4	28.6

Fig. 7a, even only 4 SQP steps and 60 ms are necessary. The benefit of reusing the path planning result as an approximation for the solution of the optimization step becomes particularly clear when comparing it with the Reeds-Shepp path (see Fig. 8). If the latter is used as an initial guess, the computation effort increases to 6 SQP steps and 120 ms for the case without moving obstacles. In the corresponding dynamic scenario, at best insufficient local minima can be identified. Thus, the proposed method leads not only to reduced processing times, but most importantly to an improved robustness.

6. SUMMARY

In this work, we have proposed algorithms for correlated motion planning and control of autonomous vehicles in dynamic urban environments. As a basis for both, an algorithm for the automated generation of free-space polygons in arbitrary situations has been presented. In particular, we have shown that this allows an efficient and generic representation of lane information and static obstacles. Subsequently, the time-dependent hybrid-state A* algorithm for model-based motion planning in non-static environments has been introduced - based on free-space polygons and dynamic Voronoi paths. We have demonstrated that it can provide short-term maneuvers in a few milliseconds even for complex scenarios, which qualifies it as a basis for making tactical decisions. Furthermore, it has been illustrated how the motion planning solution can be applied as an initial guess within a trajectory optimization step for the calculation of actual vehicle controls. Here, the comparison with Reeds-Shepp paths has shown that not only a strong reduction of the computation time is achieved, but in particular also the robustness of the optimization method for difficult situations is increased. The scope of our future work is to apply and evaluate the presented concept for decision making and control with the research vehicle from Sect. 1.2 in a real urban environment.

REFERENCES

Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars.

Büskens, C. and Wassel, D. (2012). The ESA NLP Solver Worhp. In *Modeling and Optimization in Space Engineering*, 85–110. Springer.

Dickmanns, E.D. (1997). Vehicles capable of dynamic vision. In *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence*, volume 2, 1577–1592. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. In *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*. AAAI, Chicago, USA.

Dubins, L.E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 2(3), 497–516.

Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., and Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3), 566–580.

Folkers, A., Rick, M., and Büskens, C. (2019). Controlling an autonomous vehicle with deep reinforcement learning. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE.

Knauer, M. and Büskens, C. (2019). *Modeling and Optimization in Space Engineering*, chapter Real-Time Optimal Control Using TransWORHP and WORHP Zen, 211–232. Springer Optimization and Its Applications, vol 144. Springer Verlag.

Luca, A.D., Oriolo, G., and Samson, C. (1998). Feedback control of a nonholonomic car-like robot. In *Robot Motion Planning and Control*, chapter 4. Springer.

Maurer, M., Gerdes, J.C., Lenz, B., and Winner, H. (eds.) (2016). *Autonomous Driving: Technical, Legal and Social Aspects*. Springer, Heidelberg.

Meerpohl, C., Rick, M., and Büskens, C. (2019). Free-space polygon creation based on occupancy grid maps for trajectory optimization methods. In *10th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 52, 368–374. Elsevier BV.

Paden, B., Čáp, M., Yong, S.Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1), 33–55.

Reeds, J.A. and Shepp, L.A. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.*, 145(2), 367–393.

Rick, M., Clemens, J., Sommer, L., Folkers, A., Schill, K., and Büskens, C. (2019). Autonomous driving based on nonlinear model predictive control and multi-sensor fusion. In *10th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 52, 182–187. Elsevier BV.

Schattel, A. (2018). *Dynamic modeling and implementation of trajectory optimization, sensitivity analysis, and optimal control*. PhD thesis, University of Bremen.

Shiller, Z. (2015). *Off-Line and On-Line Trajectory Planning*, volume 29, 29–62. Springer.

Siedentop, C., Heinze, R., Kasper, D., Breuel, G., and Stachniss, C. (2015). Path-planning for autonomous parking with dubins curves. In *Workshop Fahrerassistenzsysteme*.

Sommer, L., Rick, M., Folkers, A., and Büskens, C. (2018). AO-Car: transfer of space technology to autonomous driving with the use of WORHP. In *Proceedings of the 7th International Conference on Astrodynamics Tools and Techniques*.

Sydorchuk, A. (2012). The boost.polygon voronoi library.

Wolf, M.T. and Burdick, J.W. (2008). Artificial potential functions for highway driving with collision avoidance. In *2008 IEEE International Conference on Robotics and Automation*, 3731–3736.

Appendix A. HYPERPARAMETERS

TIME-SENSITIVE HYBRID-STATE A*

Descript.	Var.	Value
Weights	w_v	1
	w_o	2
	w_p	1
Voronoi	α	1000
	$d_{\max}$	4 m
Controls	$\mathcal{A}$	$\{0, \pm 1.2\}$ m²/s
	$\mathcal{B}$	$\{0, \pm 0.275, \pm 0.55\}$ rad
Discretiz.	spatial	0.5 m
	yaw	0.1 rad
	speed	0.5 m/s
	time	0.3 s

FREE-SPACE POLYGON

Descript.	Var.	Value
Refinements	τ	2
Clearance	γ	10 m
Expansion	ξ	20 m

OPTIMAL CONTROL

Descript.	Var.	Value
Weights	w_0	0.3
	w_1	0.3
	w_2	0.1
	w_3	2.5
# Discrete Points	–	31