2016 IEEE Intelligent Vehicles Symposium (IV)
Gothenburg, Sweden, June 19-22, 2016

Wiggling through complex traffic: Planning trajectories constrained by predictions

*Julian Schlechtriemen[1] , *Kim Peter Wabersich[2] and Klaus-Dieter Kuhnert[3]

***Abstract*— The vision of autonomous driving is piecewise becoming reality. Still the problem of executing the driving task in a safe and comfortable way in all possible environments, for instance highway, city or rural road scenarios is a challenging task. In this paper we present a novel approach to planning trajectories for autonomous vehicles. Hereby we focus on the problem of planning a trajectory, given a specific behavior option, e.g. merging into a specific gap at a highway entrance or a roundabout. Therefore we explicitly take arbitrary road geometry and prediction information of other traffic participants into account.**

We extend former contributions in this field by providing a flexible problem description and a trajectory planner without specialization to distinct classes of maneuvers beforehand. Using a carefully chosen representation of the dynamic free space, the method is capable of considering multiple lanes including the predicted dynamics of other traffic participants, while being real-time capable at the same time. The combination of those properties in one general planning method represents the novelty of the proposed method. We demonstrate the capability of our algorithm to plan safe trajectories in simulation and in real traffic in real-time.

I. Introduction

Autonomous driving is becoming reality. First steps can be seen in partly automated driving functions which can take over control in well-defined situations like parking, traffic jams, and highway scenarios. These (combinations of) driver assistance systems are still below of the capabilities of human drivers. Experienced human drivers understand the traffic scene which can be interpreted as the extraction of a feature vector $\boldsymbol{f}$. Based on that they predict other vehicles $\mathcal{V}$ by estimating their probable future position $p_{\mathcal{V}}(x, y, t|\boldsymbol{f})$ and use this information as constraints in combination with the static environment for decision making and trajectory-planning. In contrast to former contributions, we strongly believe that a trajectory planner should not make assumptions about the way other vehicles will continue their trajectory in the future. It has to be independent of upstream software modules and only has to be accessible using a defined geometric interface. According to this point of view our contribution covers three main aspects:

Our first contribution is an interface between scene predictions, higher level maneuver decisions and our trajectory planner based on our formal problem formulation. We want to emphasize that by construction the interface provides not only constraints but a heuristic for trajectory planning which is independent of an underlying trajectory planning method. Our second contribution covers an increased flexibility of trajectory shapes. Many sampling-based approaches lack flexibility when it comes to the shape of planned trajectories due to their limitation to a fixed number of motion primitives. Even simple scenes which require a two-step strategy, like catching up to a gap and performing a lane-change afterwards, are not directly solvable using these approaches. The third contribution is that we provide a method for algebraic optimal sampling of trajectories based on sampled trajectory points, which allows us to perform real time computations without additional effort, e.g. parallelization. More formally, given a selected behavior (e.g. a chosen gap to merge into) we formulate the trajectory planning problem as the minimization of a cost function J to compute an optimal vehicle state $\boldsymbol{x}^*(t)$, and control function $\boldsymbol{u}^*(t)$ over time[1]:

$$\boldsymbol{u}^*(t),\ \boldsymbol{x}^*(t) = \arg\min_{\boldsymbol{u},\boldsymbol{x}} J(T_f, \boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (1)$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (2)$$

$$\boldsymbol{u}(t) \in \mathcal{U} \quad (3)$$

$$\boldsymbol{x}(t) \in \mathcal{X} \quad (4)$$

$$\boldsymbol{y}(t) \in \mathcal{Y}(t) \quad (5)$$

$$t \in [0, T_f] \quad (6)$$

where $\boldsymbol{u}(t)$ is the control input of the vehicle at a time t, $\boldsymbol{x}(t)$ the vehicle state, and $\boldsymbol{y}(t)$ the vehicle position coordinates. We use a non-linear bicycle model as a reasonable simplification of the dynamical behavior of a real vehicle

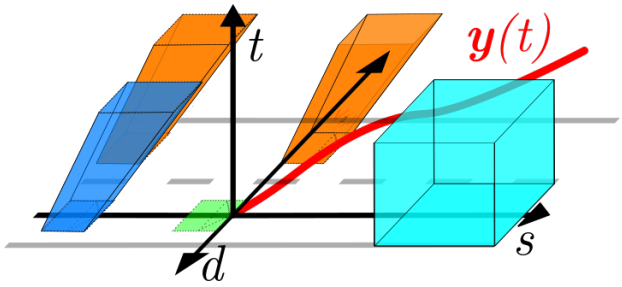


Fig. 1. Visualisation of the three dimensional planning problem. The task is to plan a safe, comfortable, and dynamic feasible trajectory $\boldsymbol{y}(t)$ for the green ego vehicle, given prediction information of other vehicles and obstacles.

[1]Julian Schlechtriemen is with Daimler Research and Development and the University of Siegen, 70134 Boeblingen, Germany julian.schlechtriemen@daimler.com

[2]Kim Peter Wabersich is with Daimler Research and Development, 70134 Boeblingen, Germany wabersich@kimpeter.de

[3]Klaus-Dieter Kuhnert is with the Institute for Real-Time Learning Systems, University of Siegen, Germany kuhnert@fb12.uni-siegen.de

*The authors contributed equally to this publication

[1]in the latter we often neglect the argument t, for example we write $\boldsymbol{x}(t)$ as $\boldsymbol{x}$.

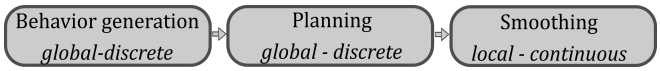


Fig. 2. We destinguish between three consecutive planning steps, in the first one a concrete behavior is chosen, e.g. to drive into a specific traffic gap. In the second a specific trajectory which is fullfilling the behavior descision is selected according to a cost-function using sampling techniques. In the third and last step the selected trajectory can be optimized locally using the assumption, that the approximate location of the minima of the cost function was determined in the previous step.

for trajectory planning purposes. Input and state constraints like maximum acceleration and velocity are incorporated, see Sec. V-A. For the definition of our cost function, see Sec. V-B. Most important, as an additional constraint, the resulting trajectory $\boldsymbol{y}(t)$ has to lie within the local constraints $\mathcal{Y}(t)$ (e.g. the road geometry, dynamic-free-space) for all time-points t up to the planning time-horizon T_f, see Sec. IV for the geometric description and Sec. V-B for the incorporation in our sampling strategy. By efficient optimal sampling (Sec. V-C) we deterministically tackle the highly non-linear optimization problem yielded by the time-varying feasible set for the output-trajectory $\boldsymbol{y}(t)$ in *real time*.

Our paper is structured as follows. In Sec. II we give the reader an overview of the current state-of-the-art methods for trajectory planning for autonomous vehicles. As a starting-point to motivate the planning problem, we shortly describe the way the occupied space can be computed as a function of t in Sec. III. In Sec. IV we describe the data representation used for the dynamic free-space, and accordingly the interface definition for our proposed trajectory planning algorithm. In Sec. V the way safe points can be sampled using the free-space information, and flexible trajectories can be generated regarding the former mentioned constraints will be described. In Sec. VI we show the capabilities of the proposed approach in simulated and real-world scenarios and conclude with Sec. VII.

II. Related work

Several methods for trajectory planning in dynamic environments have been proposed in the literature. In [1] a sampling-based method in a curvilinear coordinate system is described. The proposed method distinguishes between a finite set of maneuver options. Given those maneuvers, a fixed end-state, and the constraint to traverse exactly the sampled points, the author describes a method to plan jerk-optimal trajectories considering multiple lanes and traffic participants.

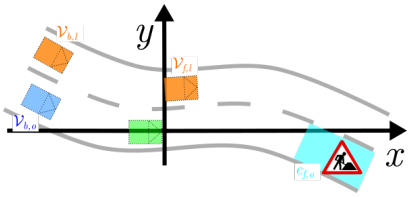
Attacking the higher-level problem as the scope of this work, the method proposed in [2] is important for our presented work. It describes a solution for the combinatorial problem of which specific maneuver variant should be chosen. For example, in the case of an overtaking maneuver and traffic in the neighboring lane, the approach is capable of selecting the best gap for the maneuver which should be executed. In the following we call this discrete selection of maneuver-variants behavior-planning, see also Fig. 2. A solution for the underlying problem of planning a concrete trajectory is described in [3]. The trajectory is planned in a global coordinate system using Sequential Quadratic Programming. This technique is able to find a local optimal trajectory along a defined centerline given a cost functional, which penalizes lateral offsets to the centerline, velocity offsets to a desired velocity, and high values of jerk and acceleration. The applicability of the method was proven on the Bertha Benz Memorial Route. The main drawback of this approach in our opinion relates to all approaches that handle the planning problem as a local optimization problem. More complex cost functions and constraints result in a real-time intractable computational effort for trajectory generation. Besides those approaches closely related to the work presented in this paper, [4] presents an approach using RRTs for dynamic motion planning in the context of autonomous driving, where the concept was proven to be feasible in the 2007 DARPA Urban challenge. Using a spatio-temporal-lattice [5] presents an approach for trajectory planning. To deal with the size of the lattice, and to deliver results in real-time, the use of a GPU is proposed, which significantly reduced the planning latency. While the computational effort, and the extensive use of memory is a drawback of this approach, the main advantage can be seen in the fact that behavior planning or the choice of maneuver variants is implicitly solved in this approach. As an orthogonal approach [6] presents a method where flexible trajectories are generated by the piecewise concatenation of motion primitives for the application of a small aerobatic helicopter.

The contribution of our work can be seen in the combination of the ideas of [1], [3], [6] and [7]. We also use a curvilinear system for sampling, while extending the areas of [1] by avoiding sampling in occupied regions. We then transform the sampled points back to the Cartesian coordinate system for trajectory generation. This guarantees driveability and jerk optimality for the generated trajectories, even when a real road contains discontinuities and curvature. By using a composite trajectory we can also improve the flexibility of the trajectory without using an overall optimization approach. Additionally our method also produces dynamically feasible results for different kinds of maneuvers, like turning at intersections, lane-changing, roundabouts, etc. because we use a longitudinal and lateral coupled model, and no path-velocity decomposition is used for trajectory generation. To generate a human-like driving behavior, we also extend the idea of [3] to penalize offsets from the center of the lane. We reformulate the idea by providing the borders of a lane as a hard constraint. Additionally the center of the lane has to be reached locally at the end of the trajectory, which enables us to realize a lateral control with extended foresight.

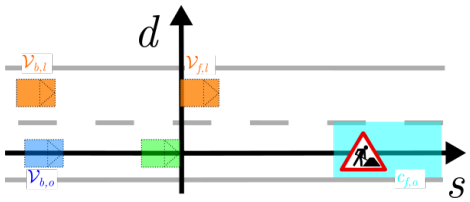
III. Estimation of Occupancies

A. Coordinate systems

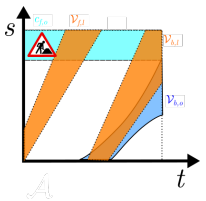
The key to planning is scene-understanding. This understanding is hard to achieve in a Cartesian coordinate system. Instead the relevance of other traffic participants to the own behavior is determined by their current and future lane-assignment. For this reason we use a curvilinear coordinate



(a) Critical situation for the green ego vehicle. How can a safe trajectory in this situation be planned?



(b) Transformation of traffic scene into curvilinear coordinate system using Γ, where the s-axis denotes the distance along the curvature of the road, and the d-axis the orthoghonal axis, compare also Fig. 3a.



(c) By using prediction information the occupied space can be visualized over the time t, see also Fig. 3b.

Fig. 3. For task of planning trajectories, we transform the traffic scene in a curvilinear coordinate system, predict other vehicles in a probabilistic fashion, and model their future positions by determining the confidence bounds for their prediction information, which is a probability density function.

system along the curvature of the road, where we denote the longitudinal dimension along the road s and the orthogonal direction d. We rely on having a function Γ which is able to transform between the vehicle coordinate system and the curvilinear system, see also Fig. 3a and Fig. 3b. The function for the transformation back to the cartesian coordinate system is called Γ^{-1} respectively:

$$P_{sd} = \Gamma(P_{xy}) \quad \text{and} \quad P_{xy} = \Gamma^{-1}(P_{sd})$$

For a short survey on how to compute this transformation see [1].

B. Prediction information

To compute the prediction information of other traffic participants, we rely on previous work [8], [9], [10]. The proposed approach is a two-step process. In the first step the probability of each maneuver-class which can be executed by other traffic participants is estimated. Because all our prediction-experiments were conducted on highway scenarios, we only distinguish between the three maneuvers lane-following Flw, lane-changing to the left LcL, and lane changing to the right LcR. For rural roads, intersections, and roundabouts more maneuver classes would be necessary. Using a Naive Bayesian approach, the probability of a vehicle with the measured feature vector $\boldsymbol{f} = \{f_1, f_2 ... f_n\}$ executing a maneuver $m \in \{Flw, LcL, LcR\}$ can be computed by:

$$p_m = p(m|\boldsymbol{f}) \propto p(\boldsymbol{f}|m)p(m), \tag{7}$$

$$p(\boldsymbol{f}|m) = \prod_{i}^{n} p(f_i|m). \tag{8}$$

Alternatively the probability p_m of a maneuver m can be computed by other machine learning techniques, see [10], [11]. Using the probability estimates of the executed maneuver class, the future positions are estimated using regression techniques. Because different drivers act in a different manner, we propose an algorithm to estimate a probability density of future positions X instead of a deterministic position estimate. For this regression problem we propose the use of a Gaussian Mixture Regression algorithm. The learning process in this case is the estimation of a Gaussian Mixture Model over the in- and output-dimensions with a distribution:

$$p(X) = \sum_{k=1}^{n} w_k \mathcal{N}(\mu_k, \Sigma_k, X) \tag{9}$$

where w_k is the weight, μ_k the mean, and Σ_k the covariance matrix of each of the n Gaussian components of the mixture. The computation of the outputs, which are the parameters of a Gaussian Mixture Distribution in this case, is solved by computing the conditional distribution, where the mean $\mu_{k,o|i}$ of a component k with the the output dimensions o given the input dimensions i and can be computed by

$$\mu_{k,o|i} = \mu_{k,o} + \Sigma_{k,o,i}\Sigma_{k,i}^{-1}(I - \mu_{k,i}). \tag{10}$$

The corresponding covaricance matrix is computed by

$$\Sigma_{k,o|i} = \Sigma_{k,o} - \Sigma_{k,o,i}\Sigma_{k,i}^{-1}\Sigma_{k,i,o}, \tag{11}$$

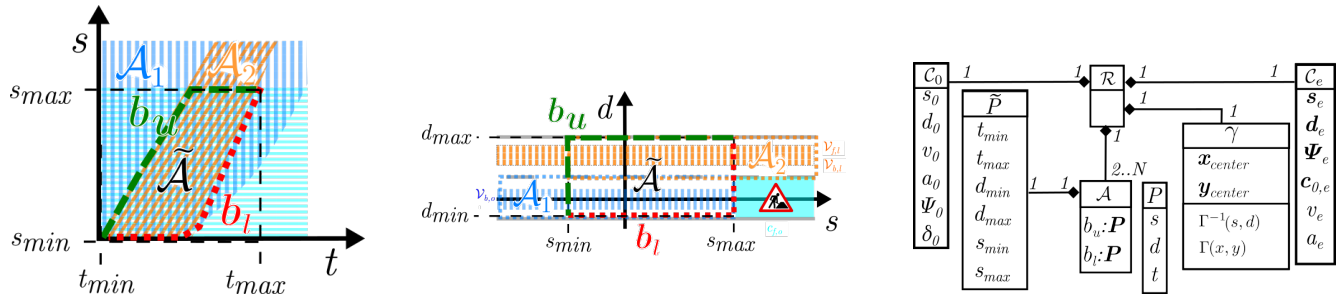
and the conditional weights by

$$w_{k|i} = \frac{w_k p(I|\mathcal{N}(\mu_k, \Sigma_k))}{\sum_{n=1}^{k} w_n p(I|\mathcal{N}(\mu_n, \Sigma_n))}. \tag{12}$$

We are using five regression models to solve the problem of position prediction of other traffic participants in highway scenarios. Two of those models are predicting the longitudinal behavior, where the first is taking into account the vehicle in front, and the second is applied if no preceding vehicle is measured by our sensors, see [9]. The three remaining models are estimating the lateral position of the maneuvers Flw, LcL, and LcR, which are then combined using p_m, see [10] for a more detailed description. To compute the deterministic occupancies of a vehicle as a function of t, we compute the confidence bounds separately for the longitudinal and lateral probability density function, see [12] for the algorithmic description to compute those bounds. Note that estimates of those confidence bounds can also be acquired by other prediction techniques, see [9] for a more detailed survey on published methods.

IV. Definition of an Interface for the Trajectory Planner

For planning trajectories we use the prediction information computed of a traffic scene, see Fig. 3b for an example. Hereby it is essential to provide sufficient constraints for



(a) By inverting the occupancies, see Fig. 3c, we can derive safe spaces which are locally and temporal adjacent. We call those areas *Action-Spaces* $\mathcal{A}$.

(b) A sectional drawing in the $s-d$ plane shows the time dependend occupancies (dashed) and the non moving, time constant occupancies (in our case a contruction yard), compare also to Fig. 3c.

(c) This slightly simplified class-diagram shows the interface of the trajectory planner.

Fig. 4. Sectional drawings of the three-dimensional planning problem, its geometric reprensentation and the resulting modelling of the interface to the trajectory planning algorithm.

each maneuver on the one hand without losing generality in the interface definition on the other hand. By representing the scene in a sectional drawing of *t-s* we can see the (future) occupancies of each lane in Fig. 3c. In Fig. 4b the corresponding occupancies are visualized in a sectional drawing in the s-d dimension. By inverting the occupancies in Fig. 3c we can visualize the safe driving space for each lane, see Fig. 4a. As can be seen in the example, there is the need for the green ego vehicle to change lanes, to avoid crashing into the construction zone $\mathcal{C}$. We decompose the given scene into three areas, called action-spaces in the following. The first action-space $\mathcal{A}_1$ is defined by the collision-free space in the starting lane. The third $\mathcal{A}_3$ is the collision-free space in the second, orange lane. We connect both of these action spaces by $\tilde{\mathcal{A}}$, which is the space that is safe in both lanes at the same time. More formally we define $\tilde{\mathcal{A}}$ as the connection action-space generated by action-space $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ if for all Points $\tilde{P} \in \tilde{\mathcal{A}}$ the following holds true:

$$\begin{pmatrix} s_{\tilde{P}} \\ d_{\tilde{P}} \end{pmatrix} \in \mathcal{A}_i \quad \text{and} \quad \begin{pmatrix} s_{\tilde{P}} \\ d_{\tilde{P}} \end{pmatrix} \in \mathcal{A}_{i+1} \quad \text{and} \quad (13)$$

$$d_p \in \mathcal{D}_{s,t} \quad \text{with} \quad \mathcal{D}_{s,t}^{\tilde{\mathcal{A}}} = \mathcal{D}_{s,t}^{\mathcal{A}_i} \cap \mathcal{D}_{s,t}^{\mathcal{A}_{i+1}} \quad \text{s.t.} \quad (14)$$

$$\mathcal{D}_{s,t}^{\mathcal{A}} = \left\{ d_{P_{s,t}} | (P_{s_t} \in \mathcal{A}) \wedge (s_{P_{s,t}} = s) \wedge (t_{P_{s,t}} = t) \right\} \quad (15)$$

$$\text{and} \quad \forall \tilde{P} : \mathcal{D}_{s_{\tilde{P}},t_{\tilde{P}}}^{\mathcal{A}_i} \cap \mathcal{D}_{s_{\tilde{P}},t_{\tilde{P}}}^{\mathcal{A}_{i+1}} \neq \{\} \quad (16)$$

which means figuratively speaking that the connecting action-space $\tilde{A}$ is the union of $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ in the *s-d* plane at all points where they intersect in the *s-t* and touch in the *s-d*-plane. The limits of an Action Space $\mathcal{A}$ are defined by the time-point t_{min} where it is the first time when it can be entered without causing a collision. The latest time-point for leaving it can be seen in t_{max}. The extreme values in the d and s dimension can be interpreted by analogy to the t dimension. We define the geometric body of an action-space $\mathcal{A}$ by a polyline called the upper-bound $\boldsymbol{b}_u$ and a lower-bound $\boldsymbol{b}_l$, each consisting of points P, see also Fig. 4c. In addition, to solve the planning problem we define a target state C_e and a start-state C_0. The start-state consists of the position in the curvilinear coordinate system and the initial values of the bicycle-model, the longitudinal velocity v_0, the longitudinal acceleration a_0, the orientation relative to the road measured by the angle Ψ and the steering angle δ. To generate human-like behavior we model the target state for our trajectory planning problem as a polyline represented by $\boldsymbol{s}_e$ and $\boldsymbol{d}_e$ providing curvature $\boldsymbol{c}_{0,e}$ and orientation $\boldsymbol{\Psi}_{0,e}$ at each point of it. Practically this allows us to generate a trajectory ending smooth on an arbitrary curved and oriented road geometry. Additionally a target speed v_e and acceleration a_e should be reached at the end of the trajectory. As the last part of the interface we need a transformation class γ, which implements the transformations between the curvilinear and Cartesian coordinate system as described in Sec. III-A.

V. Trajectory Planning

In this section we aim to solve (1) approximately[2]. Formally (1) is a kinodynamic trajectory planning problem as described in [13]. We want to emphasize that based on prediction information, we have to deal with *time-dependent* constraints. The main *challenge* is to solve (1) in real time. As described in Sec. I, the plan of attack is as follows:

- Introduction of the physical model (Sec. V-A).
- In Sec. V-B we describe how to generate a sampling set. Each of those samples consists of a tuple of meaningful output trajectory points, that the trajectory has to pass through. Those Samples can be seen as time-landmark tuples, guiding the vehicle through $\mathcal{Y}(t, \boldsymbol{y}(t))$ (5).
- We then focus in Sec. V-C on a method for algebraic, quadratic jerk-optimal trajectory generation by neglecting inequality constraints of the system dynamics and output constraints. The resulting output trajectory must hit the time-landmark tuples described above.
- Until this stage we approximated the output inequality constraints by sampling and fully neglected the state and input constraints. In the last step (Sec. V-D) we therefore filter infeasible trajectory samples.

[2]Note that without constraint (5) we would have a classical non-linear open loop optimal control problem.

A. Vehicle model

We assume that complex vehicle dynamics will be addressed by a lower-level controller. Nevertheless the trajectory planner must provide a trajectory that is roughly drivable by the car. This means that kinematic as well as dynamics and input/state constraints need to be satisfied. Therefore we use the bicycle model for trajectory planning, which is given by

$$\dot{\boldsymbol{x}}(t) = \underbrace{\begin{pmatrix} a(t) \\ \cos(\Psi(t))v(t) \\ \sin(\Psi(t))v(t) \\ \frac{1}{L}\tan(\delta(t))v(t) \end{pmatrix}}_{f(\boldsymbol{x},\boldsymbol{u})} \tag{17}$$

where

$$\boldsymbol{x}(t) = [v(t), x(t), y(t), \Psi(t)]^T, \boldsymbol{u}(t) = [a(t), \delta(t)]^T \tag{18}$$
$$\text{and } \boldsymbol{y}(t) = \underbrace{[x(t), y(t)]}_{h(\boldsymbol{x})}$$

as depicted in Fig. 5a.

Furthermore the system (17), (18) has physical input (3) and state constraints (4) like maximum velocity or acceleration. We just assume $\mathcal{U}$ and $\mathcal{X}$ to be arbitrary, but time- and state- independent sets.

B. Efficient trajectory sampling

We describe each trajectory sample, indexed by m, uniquely with an *output trajectory point* tuple

$$\mathcal{T}^{(m)} = \{\boldsymbol{y}_1^{(m)}, \boldsymbol{y}_2^{(m)}, .., \boldsymbol{y}_i^{(m)}, .., \boldsymbol{y}_N^{(m)}\}, \tag{19}$$
$$\text{s.t. } \boldsymbol{y}_1^{(m)} = h(\boldsymbol{x}_0). \tag{20}$$

Here, $\boldsymbol{x}_0 = \mathcal{C}_0$ is the vehicle state as we start planning and therefore ensures consistency with the current vehicle state for each sample, see Fig. 4c. Efficiency in our context means that we want to come up with sparse trajectory samples in the end with high feasibility rate w.r.t. constraints (2)-(5). For generation of $\mathcal{T}^{(m)}$ first consider the action-space sequence[3] $\mathcal{A}_i$, $i = 1, 2, .., N$, visualized in Fig. 4a and Fig. 4b. Observe that the final *output* trajectory we want to generate for each sample m, defined as

$$\boldsymbol{y}^{(m)}(t) = [x^{(m)}(t),\ y^{(m)}(t)]^T, \tag{21}$$

has to pass through all intersections, i.e.

$$\exists t_i,\ i = 1, 2, ..N-1 :\ \boldsymbol{y}^{(m)}(t_i) \cap \bar{\mathcal{A}}_{\text{XY},i} \neq \emptyset \tag{22}$$

where $\bar{\mathcal{A}}_{\text{XY},i}$ denotes the i-th intersected action space

$$\bar{\mathcal{A}}_{\text{XY},i} = \mathcal{A}_{\text{XY},i} \cap \mathcal{A}_{\text{XY},i+1},\ i = 1, 2, .., N-1, \tag{23}$$

represented in XY coordinates. Pictorially speaking the intersections (23) are time-dependent gates we must traverse. Therefore *our sampling heuristic* is to first discretize the intersected action spaces $\bar{\mathcal{A}}_{\text{XY},i}$ (Fig. 5b) in order to generate output trajectory point candidates:

$$\text{TP}_i^{(m)}(p, q, r) = \begin{bmatrix} t_{i,p}^{(m)} \\ s_{i,q}^{(m)} \\ d_{i,r}^{(m)} \end{bmatrix} = \begin{bmatrix} \text{t}_{\text{min}} + p\Delta_t \\ \text{s}_{\text{min}} + q\Delta_s \\ \text{d}_{\text{min}} + r\Delta_d \end{bmatrix} \tag{24}$$

where

$$\Delta_t = (\text{t}_{\text{max}} - \text{t}_{\text{min}})/N_t, \tag{25}$$
$$\Delta_s = (\text{s}_{\text{max}} - \text{s}_{\text{min}})/N_s, \tag{26}$$
$$\Delta_d = (\text{d}_{\text{max}} - \text{d}_{\text{min}})/N_d \tag{27}$$

according to Fig. 4c. N_t, N_s, N_d define the discretization resolution. If we write in the following $t_i^{(m)}$ instead of $t_{i,p}^{(m)}$ we mean an arbitrary point w.r.t. p - same for $s_{i,q}^{(m)}$ and $d_{i,r}^{(m)}$. We generate a sample (19) by choosing one trajectory point in each intersected action space. By taking every possible combination we get the tuples (19). Note that pruning can be done. For example in order to get from $\text{TP}_i^{(m)}$ to $\text{TP}_{i+1}^{(m)}$ we have that $t_{i,p}^{(m)} \leq t_{i+1}^{(m)}$ and $s_i^{(m)} \leq s_{i+1}^{(m)}$. More complex pruning strategies are also possible like rough maximum dynamics estimates.

C. Optimal output trajectory generation based on differential flatness

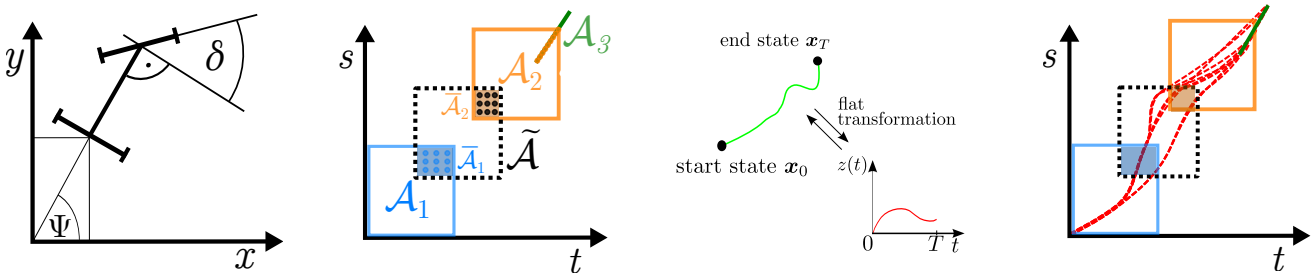
The underlying idea is to construct an output trajectory (21) that fulfills (22) such that this output trajectory is compatible with the bicycle model defined in Sec. V-A. Therefore we use the fact that the bicycle model is differentially flat w.r.t. the outputs $x(t)$ and $y(t)$. See [14] for more details. In short, if a system has the property of being differentially flat, then there exists a unique transformation between the so-called differential flat outputs $\boldsymbol{z}$ and its derivatives and the system states $\boldsymbol{x}$ and inputs $\boldsymbol{u}$, see Fig. 5c. In the case of the bicycle model we have $\boldsymbol{z} = \boldsymbol{y}$. This means that once we know an output trajectory we can calculate the system states and inputs. And vice versa: we can impose constraints on the output trajectory and its derivatives, for example at $t = 0$, such that the output trajectory corresponds to certain vehicle states and inputs.

By differentiating the so-called flat outputs $x(t), y(t)$ twice we get the mapping from the inputs $\boldsymbol{u}$ and the states $\boldsymbol{x}$ to these derivatives

$$\Phi_x : \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} \rightarrow \boldsymbol{z}_x(t) : \tag{28}$$
$$\underbrace{\begin{pmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{pmatrix}}_{\boldsymbol{z}_x(t)} = \underbrace{\begin{pmatrix} x(t) \\ \cos(\Psi(t))v(t) \\ \cos(\Psi(t))a(t) - \sin(\Psi(t))\frac{1}{L}\tan(\delta(t))v(t)^2 \end{pmatrix}}_{\Phi_x(\boldsymbol{x}(t),\boldsymbol{u}(t))},$$

$$\Phi_y : \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} \rightarrow \boldsymbol{z}_y(t) : \tag{29}$$
$$\underbrace{\begin{pmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \end{pmatrix}}_{\boldsymbol{z}_y(t)} = \underbrace{\begin{pmatrix} y(t) \\ \sin(\Psi(t))v(t) \\ \sin(\Psi(t))a(t) + \cos(\Psi(t))\frac{1}{L}\tan(\delta(t))v(t)^2 \end{pmatrix}}_{\Phi_y(\boldsymbol{x}(t),\boldsymbol{u}(t))}.$$

[3]Since we consider only single trajectory generation requests we will simply write $\mathcal{A}$ instead of $\mathcal{A}^{(k)}$.



(a) Bicycle model and coordinate system. x and y define coordinates of the center rear axis. Ψ is the angle w.r.t. an absolute coordinate frame and δ is defined as the ackerman angle.

(b) Concept of arbitrary action spaces (for simplicity just rectangular in s-t) overlapping from first (blue) to last (green) action space. Shaded areas depict intersected action spaces $\bar{\mathcal{A}}_i$ including trajectory points.

(c) Flat transformation. It enables to transform between (flat) output trajectory and state-/input trajectory. This is used in order to solve the trajectory planning problem with basis functions for the (flat) outputs between trajectory points.

(d) Visualization of trajectory samples, projected on s-t for simplicity. All the trajectories are generated using the differential flatness property in combination with jerk optimal basis functions.

Fig. 5. Basic ingredients of trajectory planning. In Fig. 5a the bicycle model, described in Sec. V-A is shown. Followed by Fig. 5b related to Sec. V-B and Fig. 5c, Fig. 5d according to Sec. V-C.

The algebraic transformation (28) and (29) can be (locally) inverted to

$$\Phi^{-1} : \begin{pmatrix} \boldsymbol{z}_x(t) \\ \boldsymbol{z}_y(t) \end{pmatrix} \rightarrow \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} : \tag{30}$$

$$\begin{pmatrix} v(t) \\ x(t) \\ y(t) \\ \Psi(t) \\ a(t) \\ \delta(t) \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{z_{x,2}(t)}{\cos[\tan^{-1}(z_{y,2}(t)/z_{x,2}(t))]} \\ z_{x,1}(t) \\ z_{y,1}(t) \\ \tan^{-1}(z_{y,2}(t)/z_{x,2}(t)) \\ \Phi_a^{-1}(.) \\ \Phi_\delta^{-1}(.) \end{pmatrix}}_{\Phi^{-1}([\boldsymbol{z}_x(t), \boldsymbol{z}_y(t)]^T)}. \tag{31}$$

where $\delta(t) = \Phi_\delta^{-1}(.)$ and $a(t) = \Phi_a^{-1}(.)$ are

$$\tan\left(\Phi_\delta^{-1}(.)\right) = \frac{z_{y,3}(t) - \tan(\Phi_\Psi^{-1}(.)) z_{x,3}(t)}{\Phi_v^{-1}(.)^2 (1/L)(\tan(\Phi_\Psi^{-1}(.)) \sin(\Phi_\Psi^{-1}(.)) + \cos(\Phi_\Psi^{-1}(.))} \tag{32}$$

$$\Phi_a^{-1}(.) = \frac{z_{x,3}(t) + \sin(\Phi_\Psi^{-1}(.))(1/L)\tan(\Phi_\delta^{-1}(.))\Phi_v^{-1}(.)^2}{\cos(\Phi_\Psi^{-1}(.))} \tag{33}$$

and Φ_v^{-1} and Φ_Ψ^{-1} are given in rows one and four of eq. (31). The inverse transformation (31) only holds locally for

$$v(t) \neq 0 \wedge \Psi(t), \delta(t) \in (-\pi/2, \pi/2). \tag{34}$$

We treat this as special cases in the implementation. Equations (28) and (29) allow us to verify state and input conditions on the output trajectory. Once an output trajectory is generated we can use equation (31) to calculate (transform) corresponding state and input trajectories.

In order to generate a trajectory for a sample $\text{TP}^{(m)}$, we split the output trajectory (21) into segments connecting $\text{TP}_i^{(m)}$ and $\text{TP}_{i+1}^{(m)}$. Consequently according to (21) we have

$$\boldsymbol{y}_i^{(m)}(t) = \begin{cases} \boldsymbol{y}_1^{(m)}(t), & t \in [t_1^{(m)}, t_2^{(m)}] \\ \vdots & \\ \boldsymbol{y}_i^{(m)}(t), & t \in [t_i^{(m)}, t_{i+1}^{(m)}] \\ \vdots & \\ \boldsymbol{y}_{N-1}^{(m)}(t), & t \in [t_{N-1}^{(m)}, t_N^{(m)}]. \end{cases} \tag{35}$$

For $\boldsymbol{y}_i(t)^{(m)}$ we introduce a trajectory piece

$$\boldsymbol{y}_i^{(m)}(t) = \begin{pmatrix} x_i^{(m)}(t) \\ y_i^{(m)}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\theta}(t)^T G_i^{(m)} \\ \boldsymbol{\theta}(t)^T H_i^{(m)} \end{pmatrix} \in \mathcal{C}^2 \tag{36}$$

based on a basis function $\boldsymbol{\theta}(t)$ with coefficients $G_i^{(m)}$ and $H_i^{(m)}$. In order to impose the constraints given by each output trajectory sample we have

$$\begin{pmatrix} x_1^{(m)}(0) \\ \dot{x}_1^{(m)}(0) \\ \ddot{x}_1^{(m)}(0) \end{pmatrix} = \Phi_x(\boldsymbol{x}_0, \boldsymbol{u}_0), \begin{pmatrix} x_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \dot{x}_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \ddot{x}_{N-1}^{(m)}(t_{N-1}^{(m)}) \end{pmatrix} = \Phi_x(\boldsymbol{x}_T, \boldsymbol{u}_T), \tag{37}$$

$$\begin{pmatrix} y_1^{(m)}(0) \\ \dot{y}_1^{(m)}(0) \\ \ddot{y}_1^{(m)}(0) \end{pmatrix} = \Phi_y(\boldsymbol{x}_0, \boldsymbol{u}_0), \begin{pmatrix} y_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \dot{y}_{N-1}^{(m)}(t_{N-1}^{(m)}) \\ \ddot{y}_{N-1}^{(m)}(t_{N-1}^{(m)}) \end{pmatrix} = \Phi_y(\boldsymbol{x}_T, \boldsymbol{u}_T) \tag{38}$$

for the start and end constraints of the whole trajectory and

(39)

$$\boldsymbol{y}_i^{(m)}(t_i^{(m)}) = \Gamma^{-1}(s_i^{(m)}, d_i^{(m)}), \qquad \forall i = 2, 3, .., N-1, \tag{40}$$

$$\boldsymbol{y}_i^{(m)}(t_{i+1}^{(m)}) = \Gamma^{-1}(s_{i+1}^{(m)}, d_{i+1,p}^{(m)}), \quad \forall i = 1, 2, .., N-2, \tag{41}$$

$$\dot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \dot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2 \tag{42}$$

$$\ddot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \ddot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2, \tag{43}$$

$$\dddot{\boldsymbol{y}}_i^{(m)}(t_{i+1}^{(m)}) = \dddot{\boldsymbol{y}}_{i+1}^{(m)}(t_{i+1}^{(m)}), \qquad \forall i = 1, 2, .., N-2. \tag{44}$$

Equations (37) and (38) impose physical vehicle start and ending constraints on the output trajectory, whereas (40) and (41) ensure that the output trajectory hits the output trajectory points. (42)-(43) yield the required smoothness conditions on the trajectory ($\mathcal{C}^2$) for the flat transformation. Finally, (43) is a consequence of the jerk optimality objective and reduces the degrees of freedom of the resulting optimization problem.

As described in the beginning of this section, we want to generate quadratic jerk-optimal trajectories. Therefore the (sub)optimization problem is to solve

$$\underset{H_i^{(m)}, G_i^{(m)},\, i=1..N-1}{\operatorname{argmin}} \sum_{i=1}^{N-1} \left(\int_{t_{i,p}^{(m)}}^{t_{i+1}^{(m)}} \left[\dddot{y}_i^{(m)}(t)^2, \dddot{x}_i^{(m)}(t)^2 \right] dt \right) \tag{45}$$

$$\text{s.t.} \qquad (37) - (44)$$

for each sample m. In [1] it is shown that the jerk-optimal output trajectory lies in quintic polynomials. Therefore we choose (36) to be quintic polynomials for $x_i^{(m)}(t)$ and $y_i^{(m)}(t)$. It follows that (45) again is a sum of polynomials and can therefore be written as a quadratic program

$$\begin{aligned} \underset{\boldsymbol{z}^{(m)}}{\operatorname{argmin}} \quad & \frac{1}{2} \boldsymbol{z}^{(m)T} Q \boldsymbol{z}^{(m)} \\ \text{s.t.} \quad & A\boldsymbol{z}^{(m)} = b \end{aligned} \tag{46}$$

where

$$\boldsymbol{z}^{(m)} = \left[H_1^{(m)T}, .., H_{N-1}^{(m)T}, G_1^{(m)T}, .., G_{N-1}^{(m)T} \right]^T. \tag{47}$$

Theorem 1: Solving (46), which is a reformulation of (45), yields a jerk-optimal trajectory $\boldsymbol{y}^{(m)}$ regarding the piecewise definition (21) with quintic polynomials as basis function.

Verification. First we verify that one must choose quintic polynomials by contradiction: Suppose each element of $\boldsymbol{y}_i^{(m)}$, $i = 1..N-1$, piecewise defined as in (21), is optimal w.r.t. quadratic jerk but elements of $\boldsymbol{y}_i^{(m)} \notin$ quintic polynomials for at least one i. By the principle of optimality [15], trajectory pieces $\boldsymbol{y}_i^{(m)}$ must be jerk optimal sequences and therefore quintic polynomials [1]. Secondly, (46) yields always global optimal state and input conditions at the output trajectory points because Q is positive semidefinite (sum of squares) and therefore (46) is convex [16].

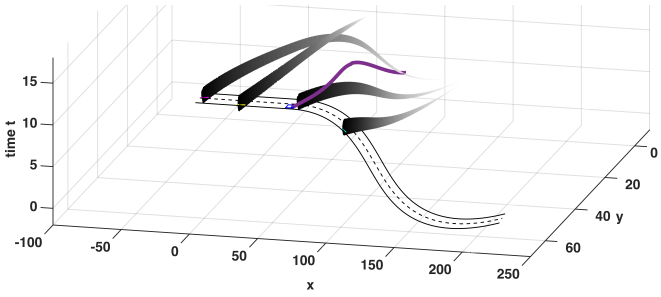


Fig. 6. Example of an output trajectory (purple) calculated for a lane change maneuver, surrounded by four vehicles. The ego vehicle is shown in blue and drives initially a bit slower than the car to the left. This scenario is difficult to solve because the ego vehicle is followed by a slower vehicle, but the ego vehicle must decelerate in order to catch the gap on the left. On the other hand, in the left lane there is also a slightly faster vehicle oncoming from behind that nearly closes the gap within the prediction horizon. The corresponding state and input trajectory is given in Fig. 7.

Algebraic solution of (45): The Lagrange function of (45) is

$$\mathcal{L}(\boldsymbol{z}^{(m)}, \boldsymbol{\nu}) = \frac{1}{2} \boldsymbol{z}^{(m)T} Q \boldsymbol{z}^{(m)} + \boldsymbol{\nu}^T (A\boldsymbol{z}^{(m)} - b). \tag{48}$$

Here, $\boldsymbol{\nu}$ are the Lagrange multipliers in vector form. Since the problem (45) is convex the Karush-Kuhn-Tucker conditions become sufficient and necessary [16]. From

$$\frac{\partial}{\partial \boldsymbol{z}^{(m)}} \mathcal{L} = 0 \Rightarrow \underbrace{\begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix}}_{A_{\text{kkt}}} \begin{pmatrix} \boldsymbol{z}^{(m)} \\ \boldsymbol{\nu} \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \tag{49}$$

with $\boldsymbol{\nu}$ Lagrange multipliers, we calculate the desired optimum $\boldsymbol{z}^{*(m)}$. With A_{kkt} having full rank we can solve (49) efficiently. Note that sampling optimal trajectories without inequality constraints does not significantly contribute to the overall runtime: for the example in Sec. VI-A, computing (49) needs less than 5% of the overall computation time. A possible (schematic) set of trajectory samples according to Fig. 5b is illustrated in Fig. 5d.

D. Choose trajectory sample

The last step of the trajectory planning is straightforward. Based on the trajectory samples we generated, see Fig. 5d, we choose the best (jerk-optimal) that satisfies all constraints. Therefore we check every trajectory generated analogue to [1].

VI. Experiments

In Fig. 6 there is a complex lane change maneuver over time depicted, where the ego vehicle is surrounded by four cars. We used four action spaces for modelling the free space and assumed constant acceleration for the neighbouring cars. We checked the constraints discretized at 50 points of a single trajectory. We shrink the action spaces based on minimum/maximum acceleration of the vehicle based on the current state, and pruned unrealistic output trajectory points out. The number of trajectories sampled depends on the

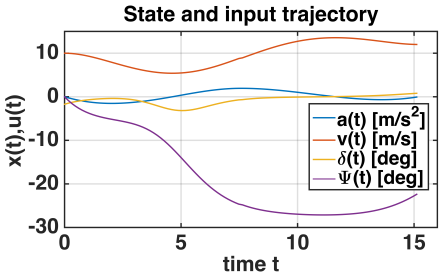


Fig. 7. Example of a trajectory calculated for a lane change maneuver, surrounded by four vehicles. The scene is depicted in Fig. 6.

density of the grid described in Sec. V-B and the pruning based on the current state. Per planning request 2000-3000 trajectories were sampled.

A. Performance

We used the EIGEN,C++ (`http://eigen.tuxfamily.org`, 18.01.16) library for efficient transformation and trajectory generation. We were able to sample around 3000 trajectories in the set-up described above within 50 ms using an off-the-shelf computer. *Note*, that we did not parallelize the computations, though it would be possible to calculate all the samples simultaneously as well as check their constraints, which is the most expensive step.

B. Synthetic test

The trajectory planning result according to the scene above is visualized in Fig. 6 and Fig. 7. The best trajectory is plotted. The resulting state and input trajectory are depicted in Fig. 7. Note how the trajectory is planned in the free spaces resulting from the predictions of the other cars, depicted with the grey tubes. Therefore the car first decelerates for the oncoming gap, goes on the left lane in the S-shaped curve, and again accelerates to the target velocity. Due to jerk optimality the acceleration profile in Fig. 7 is very smooth and the full planning time is used for accelerating to the target velocity.

C. Real world experiment

We successfully tested the concept of the trajectory planner described above in a real world experiment using the platform described in [3]. Despite its being usually hard to test dynamic scenarios in real world experiments, we set up a typical merge scenario where two busy lanes are merging. It turned out that the flexibility in how to state the planning problem using our interface comes in very handy.

VII. Conclusion

In this paper we presented a method for abstracting dynamic objects and static obstacles as time-dependent geometric bodies allowing us to develop a flexible description of maneuver problems. This enables us to describe arbitrary trajectory planning problems in structured dynamic environments. The description has been carefully chosen to provide all required information for complex maneuvers without losing generality and simplicity. It turned out that providing the constraints as a sequence of action-spaces is the key for efficient solutions of the real-time trajectory planning problem. This novel concept encapsulates a concrete behavior decision from the problem of planning a geometric trajectory. By using this description in terms of planning we presented a trajectory planner that is flexible and highly efficient compared to former approaches. This was achieved by efficient sampling of jerk-optimal trajectories that are piecewise defined in Cartesian coordinates. Future work will be on techniques for behavioral decisions as well as more flexibility in the objective of the trajectory generation.

Acknowledgment

The authors would like to thank all colleagues at MBRDNA and Daimler R&D Germany for their support and especially Qi Chen for fruitful discussions.

References

[1] M. Werling, J. Ziegler, K. Sören, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 987–993.

[2] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015, pp. 1386–1392.

[3] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for berthaa local, continuous method," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 450–457.

[4] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[5] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4889–4895.

[6] E. Frazzoli, M. Dahleh, E. Feron *et al.*, "Maneuver-based motion planning for nonlinear systems with symmetries," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1077–1091, 2005.

[7] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 250–256.

[8] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 108–114.

[9] J. Schlechtriemen, A. Wedel, G. Breuel, and K.-D. Kuhnert, "A probabilistic long term prediction approach for highway scenarios," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 732–738.

[10] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K.-D. Kuhnert, "When will it change the lane? a probabilistic regression approach for rarely occurring events," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015, pp. 1373–1379.

[11] P. Kumar, M. Perrolaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *Intelligent Vehicles Symposium (IV), 2013*. IEEE, 2013, pp. 797–802.

[12] M. Á. Carreira-Perpiñán, "Mode-finding for mixtures of gaussian distributions," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1318–1323, 2000.

[13] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

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

[15] R. Bellman, "The theory of dynamic programming," DTIC Document, Tech. Rep., 1954.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.