MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Positive Invariant Sets for Safe Integrated Vehicle Motion Planning and Control

Berntorp, Karl; Danielson, Claus; Weiss, Avishai; Di Cairano, Stefano; Erliksson, Karl; Bai, Richard

TR2019-086 September 05, 2019

Abstract

This paper describes a method for real-time integrated motion planning and control aimed at autonomous vehicles. Our method leverages feedback control, positive invariant sets, and equilibrium trajectories of the closed-loop system to produce and track trajectories that are collision-free with guarantees according to the vehicle model. Our method jointly steers the vehicle to a target region and controls the velocity while satisfying constraints associated with future motion of surrounding obstacles. We develop a receding-horizon implementation of the control policy and verify the method in both a simulated road scenario and an experimental validation using a scaled mobile robot with car-like dynamics using only onboard sensing. The results show that our method generates dynamically feasible and safe (i.e., collision-free) trajectories in real time, and indicate that the proposed planner is robust to sensing and mapping errors.

Transactions on Intelligent Vehicles

Positive Invariant Sets for Safe Integrated Vehicle Motion Planning and Control

Karl Berntorp, Richard Bai, Karl F. Erliksson, Claus Danielson, Avishai Weiss, and Stefano Di Cairano

***Abstract*—This paper describes a method for real-time integrated motion planning and control aimed at autonomous vehicles. Our method leverages feedback control, positive invariant sets, and equilibrium trajectories of the closed-loop system to produce and track trajectories that are collision-free with guarantees according to the vehicle model. Our method jointly steers the vehicle to a target region and controls the velocity while satisfying constraints associated with future motion of surrounding obstacles. We develop a receding-horizon implementation of the control policy and verify the method in both a simulated road scenario and an experimental validation using a scaled mobile robot with car-like dynamics using only onboard sensing. The results show that our method generates dynamically feasible and safe (i.e., collision-free) trajectories in real time, and indicate that the proposed planner is robust to sensing and mapping errors.**

I. INTRODUCTION

The introduction of anti-lock braking systems (ABS) and electronic stability control (ESC) in 1978 and 1995, respectively, were among the first instances of active safety systems in road vehicles [1], [2]. Several recent advanced driver-assistance and autonomous features such as lane keeping, automated lane changes, and fully autonomous driving, need control algorithms capable of generating and subsequently tracking time-varying reference trajectories over extended time periods, achieving obstacle avoidance within the operational constraints imposed by the vehicle and traffic rules.

A vehicle equipped with autonomous driving features includes a variety of different sensing and control components. Fig. 1 provides a high-level schematic for an autonomous driving system. The sensing and mapping module uses various sensor information, such as radar, lidar, camera, and global positioning system (GPS), together with prior map information to estimate the parts of the environment relevant to the driving scenario. The motion-planning module is responsible for producing a desired trajectory that the vehicle-control subsystem should follow based on the sensing-module outputs [3], [4].

Trajectory generation is often performed using either sampling-based methods such as rapidly-exploring random trees (RRTs) [5]–[10] or graph-search methods [11], [12]. Trajectory tracking is frequently done using classical control (e.g., PID) or more advanced algorithms (e.g., model-predictive control (MPC) [13]–[15]). Viewing the trajectory generation and tracking problems as decoupled, as in Fig. 1, is appealing because it simplifies the design. This is the dominant approach in the robotics community [5] and is also frequently exploited in automotive [3]. However, the time scales, dynamics, and stringent performance and driving requirements that are present in automotive systems suggests a more integrated approach to planning and control than in traditional robotics. In general, it is advantageous to consider planning and control as interacting components, and an important question is how to connect motion planning and vehicle control to ensure vehicle performance and safety [14], [16].

The authors are with Mitsubishi Electric Research Laboratories (MERL), 02139 Cambridge, MA, USA. Email: karl.o.berntorp@ieee.org

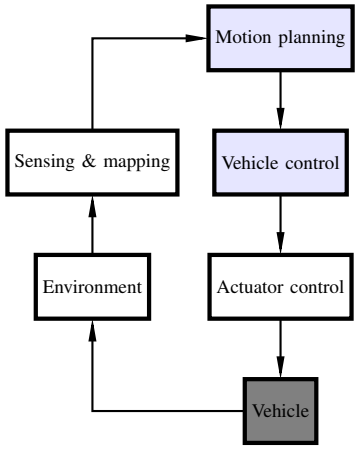

Fig. 1. A high-level system architecture of an autonomous vehicle. The different blocks can be interconnected in various ways but the main building blocks remain the same.

In this paper, we develop a motion-planning and control method that enables a wide set of maneuvers to be performed on road networks. Our method constructs partially overlapping regions in the state space, where the regions are constructed based on the system dynamics, input constraints, and state constraints (e.g., obstacles and road boundaries). Each region is a constraint-admissible positive invariant (PI) set and has an associated controller, designed together with the region, that stabilizes the system to the center of the region. We construct a graph from the regions, where graph edges are determined from the overlap of the PI sets. Then, a discrete graph search produces the sequence of nodes that provides the selection of the time-varying target points and the associated controllers. The switching of the target points and controllers according to the overlap in the PI regions provides the actual vehicle trajectory, which, because of the PI properties, is guaranteed to; (i) satisfy the closed-loop dynamics of the system; (ii) satisfy the state and input constraints; and (iii) be obstacle-free by design. We exploit a receding-horizon implementation, which provides feedback both in the planning stage and in the vehicle-control stage: obstacle avoidance and constraint-satisfaction are accounted for by design and the state-feedback control compensates for modeling errors

and provides robustness with a quantifiable margin. There is no need to perform pointwise collision checking on the generated trajectory to evaluate its safety, which implies that the computation of trajectories with the proposed method is fast, that is, orders of magnitude faster than real time on a standard laptop. This enables the implementation in automotive grade embedded platforms, which have more limited capabilities than standard caomputers [17], and the usage in rapidly changing and reactive scenarios.

A. Previous Work

In [18], we assumed a constant longitudinal velocity over the planning horizon and we only validated the approach in simulation. Our work [19] extended [18] to motion planning with time-varying longitudinal velocities. Similar to [18], [19], in this paper we formulate the planning and tracking problem in error coordinates of the vehicle with respect to the road-aligned coordinate frame. In this way, we reduce the dimensionality of the graph-search problem to an extent that computations become suitable for real-time execution. We extend [19] by providing a much more complete derivation, a more rigorous motivation for the approach, and an experimental validation showing that our method is capable of safe motion planning in presence of sensing, estimation, and modeling uncertainties. The experimental validation is done using a scaled testbench on a number of different scenarios representing different driving conditions.

Compared to methods for lane-change maneuvers and overtaking in automotive based on MPC (e.g., [20]–[22]), and algorithms connecting MPC and set invariance for obstacle avoidance in robotic systems (e.g., [23]–[26]), our method aims at reducing the computational burden to enable real-time computation even in current, or near future, automotive microcontrollers, that are significantly less powerful even than standard computers [17], while still guaranteeing safety under assumptions that are reasonable in the automotive practice. The key ingredients to achieve these conflicting objectives are our construction and usage of the invariant sets, which enable to move most of the computations offline, while demanding only lightweight computations at runtime. On the other hand, methods like MPC must rely on solving constrained optimization problems in real time, which can be computationally demanding when there are limited computational resources. For instance, [23] uses ideas from tube-based MPC to keep the state within invariant tubes by utilizing set of sum-of-squares (SOS) programming, where the motion-planning problem is assumed solved by an external motion planner.

Related ideas to our approach have been explored in [9], [27], that start from trajectories that must be feasible with respect to differential constraints, i.e., the vehicle dynamics, in the state and input space. These approaches solve SOS problems in order to obtain invariant sets around the trajectories, where [27] uses the SOS to minimize the size of the worst case reachable set due to uncertainty in the dynamics and bounded disturbances, the so called minimum positive invariant. Online, [27] solves quadratically constrained quadratic programs to guarantee a composition of collision-free funnels. The approach that we propose in this paper can be seen as the dual of those in, for example, [9], [27]. Rather than starting with obstacle free and dynamically feasible trajectories, which are 1-dimensional manifolds, and then combining them by the invariant set to become full dimensional sets used to cover the entire space, we first cover the space by invariant sets, which are full dimensional sets, and select a sequence of them by graph search, obtaining a 1-dimensional, dynamically feasible, and obstacle free trajectory as a consequence.

Alternative methods [28]–[30] assume the existence of an external trajectory generation module, and use reachability analysis for verifying whether the trajectory can be safely executed by the vehicle with respect to obstacles and traffic rules, while explicitly accounting for the model errors due to linearization and the uncertainty and external disturbances, hence accepting/rejecting the trajectory.

A distinction between reachable sets and invariant sets is that reachable sets are time varying, that is, they indicate the state reached at a certain time, and as the time changes the set changes. This may imply the need to compute (and possibly store) different reachable sets for any different time. In contrast, invariant sets are immutable over time. Hence, only computation and storage of one set is enough, which can be also carried out offline. In fact, [29] reports a ratio between real-time and computing time on a dedicated laptop of $\nu = 1.79$ for verifying safety of a previously generated trajectory (i.e., without accounting for the trajectory generation time). In contrast, the approach proposed in this paper, in similar conditions generates safe trajectories with $\nu > 100$, which allows us to consider reactive scenarios and implementation on automotive-grade electrical control units (ECUs). The work in [29] explicitly accounts for linearization errors and uncertainties, while we rely on the robustness margin achieved by the controller, for maintaining the PI set invariant under disturbances, which can be analyzed at design time.

Notation: We denote vectors in lower-case bold font as $\boldsymbol{x}$, x_j denotes the jth element of $\boldsymbol{x}$, and $\hat{\boldsymbol{x}}$ denotes the estimate of $\boldsymbol{x}$. Matrices are denoted with with $\boldsymbol{X}$ and $\boldsymbol{X}_j$ denotes the jth row of $\boldsymbol{X}$. A set $\mathcal{O}$ is PI for the system $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k)$ if $\forall \boldsymbol{x} \in \mathcal{O}$, $\boldsymbol{f}(\boldsymbol{x}) \in \mathcal{O}$. A PI set $\mathcal{O}$ is constraint admissible with respect to the constraint set $\mathcal{Y}$ if $\mathcal{O} \subseteq \mathcal{Y}$. If $V(\boldsymbol{x})$ is a Lyapunov function for the stable system $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k)$, then any level set $\mathcal{O} = \{\boldsymbol{x} \in \mathbb{R}^n : V(\boldsymbol{x}) \leq \rho\}$ is PI since $V(\boldsymbol{f}(x)) \leq V(\boldsymbol{x})$, and we write $\boldsymbol{x}_{0:k} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k\}$. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of vertices $\mathcal{V}$ and a list of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Two vertices $i, j \in \mathcal{V}$ are adjacent if $(i, j) \in \mathcal{E}$. A path is a sequence of adjacent vertices. Two vertices $i, j \in \mathcal{E}$ are connected if there exists a path connecting them.

Outline: Sec. II presents the vehicle model, system constraints, and problem definition. Sec. III describes our proposed approach, which is followed by a simulation study in Sec. IV. Sec. VI contains an experimental evaluation. Finally, Sec. VII concludes the paper.

II. Modeling and Problem Statement

We refer to the automated vehicle as the ego vehicle (EV), whereas other moving entities in the region of interest (ROI)

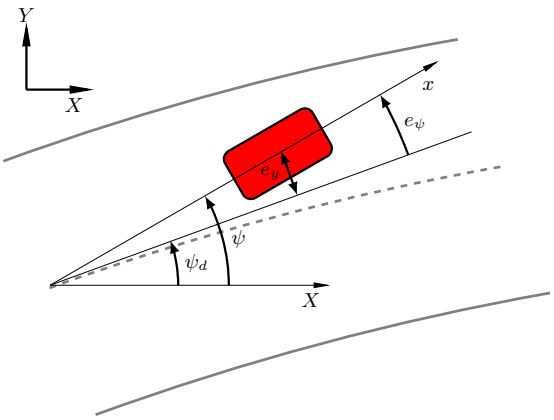

Fig. 2. Definition of the coordinate frames and related notation.

of the EV are denoted by other vehicles (OV). The OVs can be autonomous or manual. The modeling of the EV is in the local error coordinates with respect to a road-aligned frame (Fig. 2), with the origin at the start of each planning step fixed to the road center. We introduce the following assumptions.

Assumption 1: Positions and velocities of the OVs relative to the EV at the beginning of the planning phase are known.

In practice, Assumption 1 can be satisfied by measuring or estimating OV positions and velocities using onboard sensors, such as cameras, lidars, radars, and/or ultrasound sensors, or even vehicle-to-vehicle communication of GNSS data. Such an estimator may also provide an estimate of the uncertainty that could be used in the algorithm. The future states of the OVs over the planning horizon are not assumed to be known a priori but are estimated by a prediction module, see Sec. II-B. If present, the estimate of the uncertainty can be used in the traffic prediction. Sec. VI provides an experimental evaluation of how the method performs when Assumption 1 is violated.

Assumption 2: The road geometry, number of lanes, and the direction of travel in each lane is known.

Assumption 2 is usually satisfied by determining the involved quantities over the ROI by maps and onboard cameras. For instance, the lane markers can be determined by onboard cameras, as is already done today in some high-end production vehicles. This can also be complemented with map information, for example, from a car-navigation system that provides static road information such as the number of lanes and direction of travel.

A. Vehicle Model

We introduce an assumption on the driving behavior.

Assumption 3: The planner performs highway or urban driving with speeds large enough such that nonlinear effects from the powertrain are negligible, with steering action small enough that the nonlinear part of the tire-force curve is not excited, and with accelerations smooth enough that the longitudinal slip is negligible. Aggressive maneuvers, such as emergency braking and evasive steering, are handled by a separate control system.

Due to Assumption 3, the lateral dynamics are well represented by a planar single-track model with lumped right and left wheels on each axle, see Fig. 3, where the lateral tire forces are well approximated by the linear functions

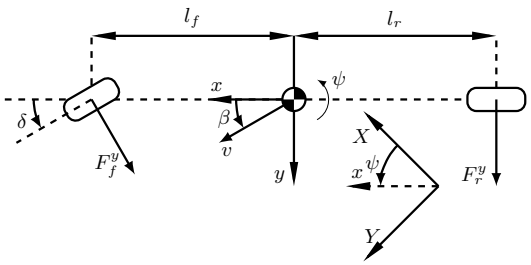

Fig. 3. A schematics of the single-track model and related notation.

$$F_{y,f} \approx C_f \alpha_f, \quad F_{y,r} \approx C_r \alpha_r, \tag{1}$$

where f, r stand for front and rear, respectively, and C_f, C_r are the front and rear lateral tire stiffness coefficients. The slip angles α_f, α_r can be approximated as [31]

$$\alpha_f \approx \delta - \frac{v_y + l_f \dot{\psi}}{v_x}, \quad \alpha_r \approx \frac{l_r \dot{\psi} - v_y}{v_x}, \tag{2}$$

where δ is the steering angle of the front wheel (a control input), v_x and v_y are the longitudinal and lateral velocity of the vehicle, respectively, $\dot{\psi}$ is the yaw rate, and l_f, l_r are the distances between the mass center and the front and rear wheel base, respectively. We introduce the lateral dynamics state

$$\boldsymbol{x}_{\text{lat}} = \begin{bmatrix} e_y & \dot{e}_y & e_\psi & \dot{e}_\psi \end{bmatrix}^{\text{T}}, \tag{3}$$

where e_y and $e_\psi = \psi - \psi_d$ denote the lateral position and vehicle orientation, respectively, in the road-aligned frame, and ψ_d is the angle of the tangent of the road with respect to the inertial frame, as defined in Fig. 2.

Using (1), (2), the model of the lateral dynamics in the error coordinates (3) can be written as the linear model [32], [33]

$$\dot{\boldsymbol{x}}_{\text{lat}} = \boldsymbol{A}_e \boldsymbol{x}_{\text{lat}} + \boldsymbol{B}_e \delta + \boldsymbol{D}_e \dot{\psi}_d. \tag{4}$$

From Assumption 3, the effect of the longitudinal slip on the tire force is negligible compared to the actuation system. Similar to standard cruise control modules, we model the response of the longitudinal acceleration $\dot{v}_x$ due to the commanded acceleration u_a as a first-order system from input torque τ to acceleration with time constant T_c, which gives the longitudinal dynamics

$$\dot{e}_x = v_x - v_{x,\text{nom}}, \tag{5a}$$

$$\dot{v}_x = -\frac{T_c}{m} v_x + T_c u_a, \tag{5b}$$

where e_x is the longitudinal position in the road-aligned coordinate frame, with respect to a nominal reference path with nominal velocity $v_{x,\text{nom}}$, m is the vehicle mass, and u_a is the control input. The desired velocity $v_{x,\text{nom}}$ is assumed constant over the planning horizon but v_x is allowed to vary.

The combined continuous-time nonlinear lateral and longitudinal dynamics are described by (4) and (5), with input vector $\boldsymbol{u} = [\delta \ u_a]^{\text{T}}$. Next, we convert the continuous-time

dynamics to discrete time with sampling period Δt, which results in the system

$$\boldsymbol{x}_{\mathrm{lat},k+1} = \boldsymbol{A}(v_{x,k})\boldsymbol{x}_{\mathrm{lat},k} + \boldsymbol{b}\delta_k + \boldsymbol{d}(v_{x,k})d_k, \quad (6a)$$

$$\boldsymbol{x}_{\mathrm{lon},k+1} = \boldsymbol{F}\boldsymbol{x}_{\mathrm{lon},k} + \boldsymbol{g}u_{a,k} + \boldsymbol{h}(v_{x,k})w_k, \quad (6b)$$

$$\boldsymbol{y}_{\mathrm{lat},k} = \boldsymbol{C}\boldsymbol{x}_{\mathrm{lat},k}, \quad (6c)$$

$$\boldsymbol{y}_{\mathrm{lon},k} = \boldsymbol{E}\boldsymbol{x}_{\mathrm{lon},k}, \quad (6d)$$

where $\boldsymbol{x}_{\mathrm{lon}} = [e_x \ v_x]^{\mathrm{T}}$, k is the time index corresponding to sampling instant t_k, $d = \dot{\psi}_d$ is the disturbance term on the lateral dynamics, and $w = v_{x,\mathrm{nom}}$ is the nominal velocity that the vehicle should track. The output equations (6c) and (6d) model the vehicle position and velocity for which the trajectory is planned, and subsequently tracked, where $\boldsymbol{y}_{\mathrm{lat},k}$ is the output vector for the lateral dynamics (6a) and $\boldsymbol{y}_{\mathrm{lon},k}$ is the output vector for the longitudinal model (6b).

As standard in discrete-time control [34], the sampling period Δt is chosen such that the discrete-time model (6) captures the relevant frequencies of the continuous-time dynamics. In addition, in this paper we ensure safety pointwise in time at sampling instants in terms of both vehicle constraint satisfaction and obstacle avoidance. Thus, Δt must be chosen such that no relevant constraint violation or collision may occur during the intersampling without appearing also at a sampling instant.

Remark 1: The term $\dot{\psi}_d$ acts as a first-order disturbance on the vehicle dynamics and arises because we model the vehicle motion in the noninertial road-aligned frame. Similar to [21], [32], [33], we ignore higher-order effects. We consider maneuvers with moderate steering and velocity changes over the planning horizon, for which this approximation has been assessed as reasonable. For high-performance maneuvering, such as during emergency collision avoidance, higher-order terms may be needed [35]. Note that $\dot{\psi}_d$ is not considered known, but estimated during driving as discussed in Sec. III-D.

B. System Constraints

To ensure that the motion planner produces trajectories that satisfy vehicle limitations and traffic rules, including obstacle avoidance and driving on the road, we impose various constraints on the vehicle states and inputs. The input vector is subject to symmetric constraints

$$-\delta_{\max} \leq \delta_k \leq \delta_{\max}, \quad (7a)$$

$$-a_{x,\max} \leq u_{a,k} \leq a_{x,\max}, \quad (7b)$$

which can be expressed as

$$\mathcal{U} = \{\boldsymbol{u}_k : \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}\}. \quad (8)$$

Constraint (8) is determined by the physical limitations of the vehicle, and for ensuring that the assumptions made for deriving (4) hold, or determined as a tradeoff between the allowed level of aggressiveness and driving comfort.

The output $\boldsymbol{y}_k = [\boldsymbol{y}_{\mathrm{lat},k} \ \boldsymbol{y}_{\mathrm{lon},k}]^{\mathrm{T}}$ is constrained as $\boldsymbol{y}_k \in \mathcal{Y}_k \in \mathbb{R}^m$, where the output set $\mathcal{Y}_k$ in general can be time-varying and is determined from different constraints. The road boundaries in the road-aligned frame impose constraints

$$-e_{y,\max} \leq e_{y,k} \leq e_{y,\max} \quad (9)$$

on the lateral position of the EV. The term $\dot{\psi}_d$ associated with the curvature of the road in the global frame, together with bounds on the allowed lateral accelerations, gives

$$\dot{e}_{\psi,\min} \leq \dot{e}_{\psi,k} \leq \dot{e}_{\psi,\max}. \quad (10)$$

Limitations on local lateral velocity error can also be set,

$$\dot{e}_{y,\min} \leq \dot{e}_{y,k} \leq \dot{e}_{y,\max}. \quad (11)$$

The constraints (9)–(11) can compactly be written as

$$\mathcal{Y}_k = \{\boldsymbol{y}_k : \boldsymbol{H}_k\boldsymbol{y}_k \leq \boldsymbol{k}_k\} \quad (12)$$

for appropriately defined $\boldsymbol{H}_k$ and $\boldsymbol{k}_k$. In this paper, (12) only refers to the lateral dynamics, since the longitudinal dynamics (6b) are completely governed by the input constraint (8) and the nominal velocity $v_{x,\mathrm{nom}}$, which is a known parameter, as will be described in detail in Sec. III. The velocity is a setpoint and the overshoot when converging to the setpoint can be adjusted by tuning the controller response to the dynamics (5). In general (12) can be time varying. However, this increases the computational burden since the computation of the regions for the planner depends on (12) and will need to be adjusted in real time. In this paper (12) is time invariant since we use a feedforward term in the controller to cancel the effects of the curvature. Hence, (12) can be determined offline.

The spatial extent of the collision area of the EV around the jth OV is denoted with $\mathcal{B}^j$. This area may depend on the geometry of the EV and may include additional safety margins as commonly done in robotics applications. The longitudinal and lateral position and velocity of each OV relative to the EV are included in the state vector $\boldsymbol{x}_{\mathrm{OV}}$ and add further time-varying constraints on the outputs of the EV. We define the jth obstacle set at time step k as $\mathcal{D}(\hat{\boldsymbol{x}}^j_{\mathrm{OV},k}, \mathcal{B}^j)$, which is a function of the predicted OV state vector and the spatial extent. In this way we can also model both deterministic and probabilistic (i.e., uncertain) regions for the OV spatial extent. Denote the planning horizon with N_p. Then, the predicted set of the jth obstacle for each $k \in [0, N_p]$ is

$$\mathcal{S}^j_k = \mathcal{D}(\hat{\boldsymbol{x}}^j_{\mathrm{OV},k}, \mathcal{B}^j). \quad (13)$$

The motion-planning method proposed in this paper is compatible with various obstacle-prediction and threat-assessment methods proposed in literature [36]. The collision-avoidance area at time index k is the union of all OV trajectory sets (13),

$$\mathcal{S}_k = \bigcup_{j=1}^{M} \mathcal{S}^j_k. \quad (14)$$

The set (13) can be probabilistic, such as an uncertainty region computed with statistical methods [37]. A trajectory is collision free as long as the PI sets associated with the trajectory do not intersect (13).

C. Problem Statement

The objective of the integrated motion-planning and vehicle control approach developed in this paper is to generate an input trajectory $\boldsymbol{u}_k \in \mathcal{U}$ over the planning horizon N_p, $k \in [0, N_p]$, such that the resulting trajectory obtained from (6)

satisfies the constraints (12) and avoids the obstacle set (14) pointwise in time, for all $k \in [0, N_p]$, and reaches a given goal region $\mathcal{X}_{\text{goal}}$, that is, $\boldsymbol{x}_{N_p} \in \mathcal{X}_{\text{goal}}$.

The goal region $\mathcal{X}_{\text{goal}}$ is not required for the operation of the method and may be redundant in scenarios such as highway driving. In fact, if no viable path exists to $\mathcal{X}_{\text{goal}}$, the algorithm will still ensure safety while moving as close as possible, in a specific metric, to it. The goal region may be useful to make sure that the vehicle obeys traffic rules, such as stopping at an intersection or switching lane in preparation for a turn.

III. Safe Motion Planning Using Positive Invariant Sets

In this section we describe our method for solving the integrated motion-planning and vehicle-control problem. Our focus is safe real-time motion planning when computing resources are limited, as it happens in embedded computing platform for automotive applications [17]. This implies that the main objective is to quickly find smooth, drivable trajectories that avoid collision, rather than searching for the optimal one.

The main idea of the method is that we determine regions on the road where it is safe to travel, each region being associated with the controller that renders it invariant for the vehicle dynamics and enforces the vehicle constraints. Then, we construct a graph by determining, through the overlap of the regions, from which region we can safely move to which other region without collision and while satisfying vehicle dynamics and constraints. Finally, we compute the trajectory to navigate the road by graph search to find a safe path through the regions. This also determines the sequence of tracking controllers that generates the closed-loop trajectories.

A. Feedback Control Design

In our preliminary work [18] we generated paths by connecting equilibrium points that correspond to lateral positions on the road, where each equilibrium was associated with an invariant set. However, to allow variable velocity we must include velocity information into the equilibria.

We formulate the path-planning problem as a graph-search problem over the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of vertices $\mathcal{V}$ and edges $\mathcal{E}$ incorporating the closed-loop dynamics. The planning horizon N_p is constructed with sampling period T_s, where $T_s = \ell \Delta t$ is a multiple of the sampling period Δt for some $\ell \in \mathbb{N}$. At each time step $k \in [0, N_p]$ in the planning phase we define a set of R candidate equilibrium references in longitudinal velocity and lateral position as

$$\mathcal{R} = \{\boldsymbol{r}_k^j\}_{j=1}^R \subset \mathbb{R}^2, \tag{15}$$

where each equilibrium reference point

$$\boldsymbol{r}_k^j = \begin{bmatrix} v_{x,\text{nom}}^j & r_{y,k}^j \end{bmatrix}^{\mathrm{T}} \tag{16}$$

is modeled in the local vehicle frame relative to the position and velocity at the beginning of the planning phase. Due to the sampling period Δt of the vehicle dynamics, the nominal longitudinal relative position reference $r_{x,k}$ is also defined. The set of lateral reference points $\{r_y^i\}_{i=1}^{n_y}$ define a grid on the road and includes the middle of each lane. For each candidate nominal velocity $v_{x,\text{nom}}^j$ in the set $\{v_{x,\text{nom}}^j\}_{j=1}^{n_v}$ the relative motion of the OVs to the EV will be different. Consequently, as long as at least one of the velocities in the set $\{v_{x,\text{nom}}^j\}_{j=1}^{n_v}$ leads to a feasible trajectory, by checking for feasibility in the set of candidate reference velocities, the graph search will eventually find a collision-free closed-loop trajectory for a given reference velocity $v_{x,\text{nom}}^j$. The number of reference paths in (15) is $R = n_y n_v$ and constitute reference paths over the entire planning horizon N_p.

The motion planner integrates trajectory generation and trajectory tracking by exploiting state-feedback controllers for both longitudinal and lateral motion. The longitudinal dynamics (6b) are linear, and we want to track the nominal velocity v_{nom}. We design a state-feedback controller

$$u_{a,k} = -\boldsymbol{\kappa}_x^{\mathrm{T}} \boldsymbol{x}_{\text{lon},k} \tag{17}$$

where $\boldsymbol{\kappa}_x$ is the feedback gain. The lateral dynamics (6a) are nonlinear in v_x. We linearize (6a) about the nominal reference velocity $v_{x,\text{nom}}^j$, which gives a locally linear model

$$\tilde{\boldsymbol{x}}_{\text{lat},k+1} = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{x}}_{\text{lat},k} + \boldsymbol{b}\delta_k + \tilde{\boldsymbol{d}} d_k. \tag{18}$$

We use a local state-feedback controller with integral action [34] by adding $\epsilon_{y,k} = \epsilon_{y,k-1} + \Delta t e_y$ and augmenting the state vector as $\bar{\boldsymbol{x}}_{\text{lat}} = [\tilde{\boldsymbol{x}}_{\text{lat}}^{\mathrm{T}} \ \epsilon_y]^{\mathrm{T}}$. With such an augmentation, the reference setpoint vector becomes

$$\boldsymbol{r}_y^i = \begin{bmatrix} 0 & r_y^i & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{19}$$

which results in the controller

$$\delta_k = -\boldsymbol{\kappa}_y^{\mathrm{T}} (\boldsymbol{r}_y - \bar{\boldsymbol{x}}_{\text{lat},k}) + \delta_k^{ff} \tag{20}$$

for the augmented system. The feedforward term $\delta_k^{ff} = -\boldsymbol{b}\boldsymbol{d}^{-\dagger} d_k$, where $\dagger$ is the pseudo-inverse, corrects for the disturbance d_k due to the curvature of the road. The feedback gain $\boldsymbol{\kappa}_x$ globally asymptotically stabilizes $v_{x,\text{nom}}$ and $\boldsymbol{\kappa}_y$ locally stabilizes the lateral reference point (16). Model (18) is usually linearized for multiple reference velocities, for which a different state feedback controller is designed. Hence, the bound on the linearization errors is determined by the number of nominal velocities n_v, which is a design parameter whose effect can be analyzed offline.

B. Offline Graph Construction

To know which reference points are safe to connect, and hence how to determine a safe path between reference points when accounting for the closed-loop dynamics, we us PI sets. We construct a family of PI sets $\{\mathcal{O}_i\}_{i=1}^{n_o} \subseteq \mathbb{R}^5$ of states $\bar{\boldsymbol{x}}_{\text{lat}} \in \mathbb{R}^5$ for different nominal velocities $v_{x,\text{nom}}^j$. Each PI set $\mathcal{O}_i$ is a sublevel set of the quadratic Lyapunov function $V(\bar{\boldsymbol{x}}_{\text{lat}} - \boldsymbol{r}_y^i)$ associated with (6a) in closed-loop with the controller $\boldsymbol{\kappa}_y$. The PI sets guarantee that the closed-loop trajectory locally satisfies constraints (7a) and (12). The ith PI set is

$$\mathcal{O}_i = \{\bar{\boldsymbol{x}}_{\text{lat}} \in \mathbb{R}^5 : (\bar{\boldsymbol{x}}_{\text{lat}} - \boldsymbol{r}_y^i)^{\mathrm{T}} \boldsymbol{P} (\bar{\boldsymbol{x}}_{\text{lat}} - \boldsymbol{r}_y^i) \leq \rho_i\}, \tag{21}$$

where $\boldsymbol{P}$ is a symmetric positive definite matrix associated with the Lyapunov function $V(\cdot)$ [38]. Although each reference equilibrium point $\boldsymbol{r}_y^i$ has an associated PI set $\mathcal{O}_i$, storage-wise typically $n_o \neq n_y$ since the scale factor ρ_i is the same

for multiple invariant sets. This means that the same invariant set $\mathcal{O}_i$ can be used, in a shifted form, for multiple reference points, which reduces memory requirements. Since $V(\cdot)$ is a Lyapunov function associated with the feedback gain $\boldsymbol{\kappa}_y$, any state trajectory that is initially inside $\mathcal{O}_i$ will remain inside $\mathcal{O}_i$ for all $k > 0$ if $\boldsymbol{r}_y^i$ is unchanged. The scale factor ρ_i is determined as the largest value such that $\mathcal{O}_i$ does not violate the input constraints (7a) and the static output constraints (12). Determining ρ_i is in general a nonconvex optimization problem. However, for our system (6a) and constraints (8), (12), the problem has the closed-form solution [38]

$$\rho_i = \min_j \left\{ \frac{\delta_{\max} - \delta_k^{ff}}{\|\boldsymbol{\kappa}_y^{\mathrm{T}} \boldsymbol{P}^{-1/2}\|}, \frac{e_{y,\max} - r_y^i}{\|\boldsymbol{C}\boldsymbol{P}^{-1/2}\|} \right\}. \tag{22}$$

Each vertex $v \in \mathcal{V}$ of the graph for each nominal velocity $v_{x,\mathrm{nom}}^j$ includes the lateral equilibrium point r_y^i, the state-feedback controller $\boldsymbol{\kappa}_y$ that stabilizes the equilibrium point, and the safe set $\mathcal{O}_i$ associated with the state-feedback controller. The edges $\mathcal{E}$ indicate which of the setpoints are connected by safe trajectories. An equilibrium point $\boldsymbol{r}_{y,i}$ with PI set $\mathcal{O}_i$ is connected to $\boldsymbol{r}_{y,j}$ with PI set $\mathcal{O}_j$ in ℓ time steps (i.e., in one planning step) if $\mathcal{O}_i$ is contained in $\bar{\mathcal{O}}_j^\ell$ [39],

$$\mathcal{O}_i \subseteq \bar{\mathcal{O}}_j^\ell, \tag{23}$$

where

$$\bar{\mathcal{O}}_j^\ell = \{\bar{\boldsymbol{x}}_{\mathrm{lat}} \in \mathbb{R}^5 : (\bar{\boldsymbol{x}}_{\mathrm{lat}} - \boldsymbol{r}_y^j)^{\mathrm{T}} \bar{\boldsymbol{P}} (\bar{\boldsymbol{x}}_{\mathrm{lat}} - \boldsymbol{r}_y^j) \leq \rho_j\}, \tag{24}$$

where $\bar{\boldsymbol{P}} = (\bar{\boldsymbol{A}}^\ell)^{\mathrm{T}} \boldsymbol{P} \bar{\boldsymbol{A}}^\ell$, $\bar{\boldsymbol{A}} = \tilde{\boldsymbol{A}} - \boldsymbol{b}\boldsymbol{\kappa}_y$. Evaluating (23) exactly requires solving a nonconvex quadratically-constrained quadratic program. However, a sufficient condition for (23) to hold is

$$(\boldsymbol{r}_y^i - \boldsymbol{r}_y^j)^{\mathrm{T}} \bar{\boldsymbol{P}} (\boldsymbol{r}_y^i - \boldsymbol{r}_y^j) \leq \rho_j - \rho_i \left\| \boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2} \right\|_F, \tag{25}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Checking for connectivity using (25) is simple but it may be conservative. An alternative is to check

$$(\boldsymbol{r}_y^i - \boldsymbol{r}_y^j)^{\mathrm{T}} \boldsymbol{P} (\boldsymbol{r}_y^i - \boldsymbol{r}_y^j) \leq \frac{\sqrt{\rho_j} - \sqrt{\rho_i} \left\| \boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2} \right\|_2}{\left\| \boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2} \right\|_2}, \tag{26}$$

which is typically less conservative than (25) since $\|\boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2}\|_2 \leq \|\boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2}\|_F$ and the division of the right-hand side in (26) with $\|\boldsymbol{P}^{-1/2} \bar{\boldsymbol{A}} \boldsymbol{P}^{1/2}\|_2 < 1$.

With all edges between the vertices determined, we construct a weighted adjacency matrix $\boldsymbol{M}^i$ for each reference velocity $v_{x,\mathrm{nom}}^i$ between all vertices in $\mathcal{V}$. A connection between two vertices is indicated by setting the edge weight w_{ij} to the cost of moving along the edge (i, j). Edges corresponding to transitions between the middle of either of the lanes may have a low cost, whereas transitions close to the road boundaries may have larger cost. The edge weights of the connectivity graph $\mathcal{G}$ can be used as design parameters to ensure that the motion planner follows a desirable driving behavior. Ramp-like cost landscapes are constructed around OVs to make the predefined safety margins soft. This encourages the EV to stay further away from the obstacles but does not force the EV to drastically reduce its speed if the EV cannot turn as fast as the motion planner suggests.

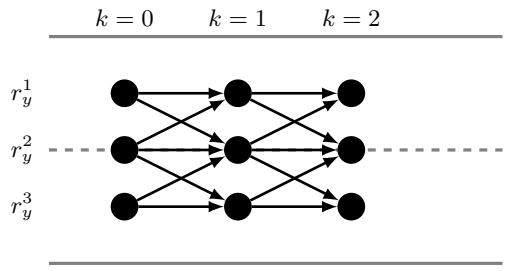

Fig. 4. An example of a small graph of three vertices (filled circles) and edges (arrows) that connect the vertices over the three time steps. The graph corresponds to the adjacency matrix in Fig. 5.

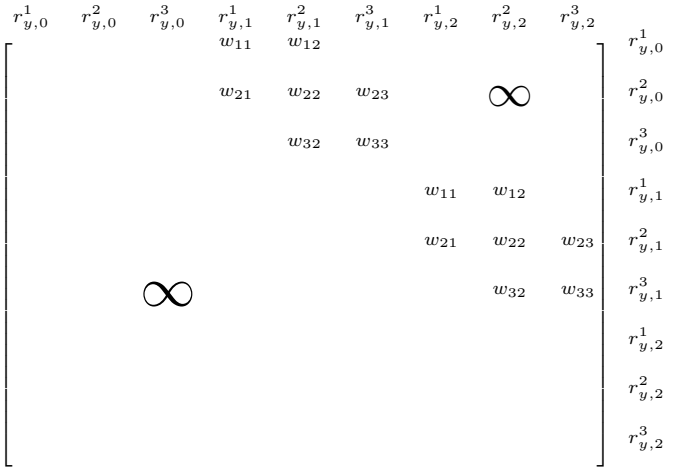

Fig. 5. The adjacency matrix corresponding to the graph in Fig. 4.

Because of time-causality and size limitations of the PI sets, the adjacency matrices $\boldsymbol{M}^i$ will be upper-block diagonal and extremely sparse. Fig. 4 shows an example graph for the case of three lateral reference points $\{r_y^i\}_{i=1}^3$ and a planning horizon $N_p = 2$. In this example, the vertices are connected with edges to the neighboring vertices. For instance, when driving in the middle of the road it is possible to move both to the left and right. However, from the outer vertices, it is only possible to switch to the closest reference point, which implies that it would take two time steps to change lane. If two vertices are not connected, the corresponding edge weight in the adjacency matrix is set to ∞. Fig. 5 displays the adjacency matrix $\boldsymbol{M}$ corresponding to the graph in Fig. 4.

In the graph construction we have a set of nominal velocities around which different linearization models are computed, and the PI sets will be different for the different nominal velocities. Indeed, due to linearization, there may be differences between the expected trajectory according to the linearized model (18) and the actual vehicle trajectory according to (6a). However, such errors will only be practically visible in the transient phase when we switch between nominal velocities, which only happens at the beginning of a planning phase.

In this regards, an advantage of the proposed method that exploits feedback and uses as PI sets the sublevel sets of local quadratic Lyapunov functions is that along the nominal trajectory the Lyapunov function decreases as $\Delta V(\bar{\boldsymbol{x}}_{\mathrm{lat}}) = -(\bar{\boldsymbol{x}}_{\mathrm{lat}} - \boldsymbol{r}_y^i)^{\mathrm{T}} \boldsymbol{Q} (\bar{\boldsymbol{x}}_{\mathrm{lat}} - \boldsymbol{r}_y^i)$, where $\boldsymbol{Q}$ is positive definite. In nonnominal conditions, each sublevel set of V is still PI

as long as $\Delta V(\bar{\boldsymbol{x}}_{\rm lat}) \leq 0$. Thus, invariance of $\mathcal{O}_i$ for the original system (6a) is still guaranteed as long as the error on the Lyapunov function $\Delta V(\bar{\boldsymbol{x}}_{\rm lat})$ due to model errors or other disturbances satisfies $\Delta V(\bar{\boldsymbol{x}}_{\rm lat}) < \bar{\boldsymbol{x}}_{\rm lat}^T \boldsymbol{Q} \bar{\boldsymbol{x}}_{\rm lat}$. This condition is actually only needed to ensure both safety and convergence, while if only safety is the concern, the condition is $\Delta V(\bar{\boldsymbol{x}}_{\rm lat}) \leq (\bar{\boldsymbol{x}}_{\rm lat} - \boldsymbol{r}_y^i)^{\rm T} \boldsymbol{Q} (\bar{\boldsymbol{x}}_{\rm lat} - \boldsymbol{r}_y^i) + (\rho - V(\bar{\boldsymbol{x}}_{\rm lat}))$. These conditions can be used to determine the allowed model error and from this, the set of nominal velocities $\{v_{x,\rm nom}^j\}_{j=1}^{n_v}$.

The size of the invariant sets will change with the velocity setpoint we linearize around to get the linear model (18). Therefore, depending on the range of reference velocities being used, several lateral controllers may need to be designed to ensure that stability and connectivity is maintained, which in the worst case gives n_v different adjacency matrices, one for each velocity setpoint. In practice, we design a lateral controller for a subset of the range of velocities such that connectivity and stability in this subset is guaranteed in a similar fashion as for the lateral setpoints within each velocity (see (24)). This reduces the amount of adjacency matrices to be stored in the, usually limited, computing platform.

In the adjacency matrix $\boldsymbol{M}$ we can encode a minimum length of the planning horizon [39]. The nominal planning horizon is N_p steps long, but it may be possible to reach a target region $\mathcal{X}_{\rm goal}$ in fewer steps, and due to the stage costs associated with traversing the nodes, the graph search will favor solutions that involve fewer steps. As short solutions may be overly aggressive for passengers, we encode in $\boldsymbol{M}$ that the path must be at least N_m steps long, where $0 < N_m \leq N_p$.

Remark 2: Quantification of the disturbances and modeling errors that can be tolerated by the proposed approach can be performed in several ways. The maximum 2-norm of the additive disturbance as percentage of the current state that can be tolerated is determined by the maximum $\rho > 0$ such that $(\bar{\boldsymbol{A}}\boldsymbol{z} + \boldsymbol{w})^{\rm T} \boldsymbol{P} (\bar{\boldsymbol{A}}\boldsymbol{z} + \boldsymbol{w}) - \boldsymbol{z}\boldsymbol{P}\boldsymbol{z} \leq -\varepsilon \boldsymbol{z}^{\rm T}\boldsymbol{z}$, for all vectors $\boldsymbol{w}$ such that $\boldsymbol{w}^{\rm T}\boldsymbol{w} \leq \rho^2 \boldsymbol{z}^{\rm T}\boldsymbol{z}$, where $\boldsymbol{z} = \bar{\boldsymbol{x}}_{\rm lat} - \boldsymbol{r}_y^i$ and $\varepsilon > 0$ is arbitrarily small, which is solved by a linear matrix inclusion (LMI) [40, (5.10)–(5.14), adapted to discrete-time]. Relating this to parameter errors is straightforward and as a consequence, one can tune the control design (and therefore the invariant sets) to be more or less conservative. For safety to be preserved, by Nagumo's Theorem [41] it is sufficient for this to hold at the border of the invariant set, that is, for all $\bar{\boldsymbol{x}}_{\rm lat}$ such that $\boldsymbol{z}^{\rm T}\bar{\boldsymbol{P}}\boldsymbol{z} = \rho$, although to account for discrete-time behavior, a "strip" around the border is more appropriate. For the design used to obtain the simulation results in Sec. IV, $\rho > 9.5\%$. In addition, for range uncertainty on the parameters, quantification amounts to feasibility checking of $(\bar{\boldsymbol{A}}\boldsymbol{w})^{\rm T}\boldsymbol{P}\bar{\boldsymbol{A}}\boldsymbol{w} - \boldsymbol{P} \leq -\varepsilon \boldsymbol{I}$, for all $\boldsymbol{w} \in \mathcal{W}$, where $\mathcal{W}$ is the range uncertainty on the parameters, which again can be solved via LMI after reformulation as a polytopic difference inclusion [40, (5.8), adapted to discrete-time].

Remark 3: In this paper we use PI sets for guaranteeing safety, relying on the robustness margin due to Lyapunov function and the related feedback against modeling errors and possible uncertainties, as discussed above. Alternatively, one may use robust PI sets (RPI) that explicitly account for additive or multiplicative disturbances during set construction, based on (nontrivial) uncertainty quantification. Given that the errors stemming from model uncertainty and linearization tend to be multiplicative (e.g., trigonometric expressions of the rotation, linear gain of stiffness, and coefficients due to velocity errors) one could compute robust feedback gains and corresponding RPI sets for a difference inclusion akin to [42]. In this paper we use PI sets since, for the application at hand, the robustness from the Lyapunov function and related feedback appears sufficient to compensate for modeling errors and uncertainties.

C. Obstacle Avoidance and Online Graph Search

The connectivity test between equilibrium points (25) is done offline and in absence of any OV. While it is possible to change online the size of ρ_i and ρ_j in (21) and (24), respectively, depending on the obstacle constraints $\mathcal{S}_k$, the computational cost tends to be too high for real-time high-frequency implementation on an embedded platform for automotive applications. Instead, we exploit again the properties of the PI sets for reducing the computation as follows. During runtime, we check for intersection of the PI sets with the obstacle set (14). If

$$\mathcal{O}_i \bigcap \mathcal{S}_k \neq \emptyset, \tag{27}$$

the equilibrium point $\boldsymbol{r}^i$ for $\mathcal{O}_i$ is marked as unsafe and the corresponding vertex (i, k) is eliminated from the graph.

We employ Dijkstra's algorithm for the graph search in our simulations and experiments. We search through the set of connectivity graphs $\boldsymbol{M}$, according to the preferred velocity order, until we find a feasible reference path. If there exists no path to the goal region, we modify the goal region so that its associated node is reachable. One approach is to execute Dijkstra's algorithm with the current node as the goal and "reverted edges" to determine which nodes are currently reachable, and then to select as modified goal the node that has the least distance to the original goal node for the graph that does not consider the obstacles.

When the graph search is completed and the reference path has been found, this path is submitted to the controllers (17), (20) for execution and subsequent real-time tracking. The preferred velocity can be adjusted during runtime. For instance, on highway driving it is reasonable to start with the speed limit and then try different velocities in decreasing order. However, when stopping at an intersection the preferred velocity is zero, possibly with a transition phase through intermediate velocities to ensure safe deceleration of the vehicle.

D. Implementation Aspects

The vehicle model (4) assumes knowledge of the disturbance $\dot{\psi}_d$, which has to be estimated. The disturbance can be written as

$$\dot{\psi}_d(t) = v_x c(t), \tag{28}$$

where $c(t)$ is the road curvature, which is an unknown function of time. However, it is possible to point-wise estimate the curvature given data points of either the road boundary or the lane markers, or from a map.

Due to sensor errors and unpredicted changes in the environment, especially at long distances, the computed steering inputs and corresponding trajectory are implemented as a receding horizon strategy. The computed trajectory is N_p steps long but is only applied for a portion N_c of the whole plan. This ensures that feedback is not only imposed during the trajectory tracking but also in the planning stage.

To account for modeling errors, assumptions not strictly holding in practical cases, and estimation errors of the OVs relative to the EV not captured by the constraint set (13), we introduce an additional safety time of N_s time steps for the longitudinal uncertainties and a lateral safety margin w for uncertainties in the lateral direction.

The parameters, n_r and T_s that indicate the number of setpoints and the sampling-period in the planner, respectively, are tightly connected. A small T_s means that a large number of setpoints n_r is needed, since the switching of the reference points is made with respect to T_s. Instead, the obstacle avoidance is guaranteed at sampling instants seperated by Δt. Thus, other than by classical factors such as capturing relevant frequencies, sensor update rates, and available computing power, Δt is selected by considering that no relevant event may occur during the intersampling which is not visible at some sampling instant.

The number of lateral reference points r_y^i is a tuning parameter that can be used to determine the number of switches and connectivity of the search graph. Increasing the number of reference points results in a larger graph and therefore a larger computational cost, but it also improves connectivity in the graph. Furthermore, a large number of reference points makes it possible to design smaller PI sets. In the offline phase this reduces connectivity since there is less overlap between the PI sets. On the other hand, in the online collision checking this can result in fewer nodes being eliminated from the graph search, thereby effectively increasing connectivity.

The motion planner updates the motion plan with an allocated computation time. Hence, the delay will cause a mismatch between the EV position at the start of the planning phase and the actual position of the EV when it receives the motion plan. To account for this mismatch, we use a delay compensator that predicts the EV position for the allocated computational time. To generate the trajectory from the determined path, we switch setpoints to the controller and subsequently generate the trajectory by forward simulation of the closed-loop dynamics between the time instants of the switching. This gives the state trajectory $\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+N_p}$.

E. Algorithm Summary

The proposed algorithm is summarized in Algorithm 1. The algorithm assumes a state estimate $\hat{\boldsymbol{x}}_0$ of the ego vehicle and the M OVs ($\{\hat{\boldsymbol{x}}_{\mathrm{ov},0}\}_{j=1}^{M}$ at the time instant corresponding to the beginning of the planning phase. If there exists no path to $\mathcal{X}_{\mathrm{goal}}$, this is detected at Line 7, and the goal is modified as previously discussed. Most of the computations are done offline. Online, the most demanding task is to perform the prediction of obstacles (Line 4) and the intersection test (Line 5). Line 5 scales linearly with the number of obstacles and the collision checks are quadratic in the output dimension. Note that the collision checks are performed only on the graph nodes to remove edges from the graph, and not on every point of the vehicle trajectories, to evaluate its safety, since safety is guaranteed by the PI sets. The graph search (Line 7) is computationally fast, since the graph matrix $\boldsymbol{M}$ is upper block-diagonal (due to causality) and sparse, and the causality decreases the complexity for solving Dijkstra's algorithm from $O(|\mathcal{E}| + |\mathcal{V}| \log(|\mathcal{V}|))$ to $O(|\mathcal{E}| + |\mathcal{V}|)$. Furthermore, applying standard state-feedback control (Line 14) is computationally inexpensive. The total computational cost depends on the number of obstacles in the region of interest and how many reference velocities (i.e., graphs) the method needs to traverse before finding a solution.

Algorithm 1 Proposed method

Offline: Compute $\mathcal{O}_i$ using (21), (22) $\forall i \in [1, \ldots, n_r]$ for different velocity setpoints and construct adjacency matrices $\boldsymbol{M}$ as needed by determining (23) using (25).
1: **Input:** $\hat{\boldsymbol{x}}_0$, $\{\hat{\boldsymbol{x}}_{\mathrm{ov},0}\}_{j=1}^{M}$, $\mathcal{X}_{\mathrm{goal}}$.
2: Predict obstacle set (14).
3: **for** $v \in \{v_{x,\mathrm{nom}}^i\}_{i=1}^{n_v}$ **do**
4: Closed-loop prediction of EV using v.
5: Check for intersection using (27) and remove corresponding edges in $\boldsymbol{M}^i$.
6: Determine r_y^i such that $\hat{\boldsymbol{x}}_0 \in \mathcal{O}_i$ from (21).
7: Perform a graph search to find a reference path $\bar{\boldsymbol{r}}_{0:N}$, $N \in [N_m, N_p]$, where $\boldsymbol{r}_N \in \mathcal{X}_{\mathrm{goal}}$.
8: **if** Solution found **then**
9: Go to Line 12.
10: **end if**
11: **end for**
12: **for** $k = 1$ **to** N_c **do**
13: Estimate $\dot{\psi}_d$ from (28).
14: Control the vehicle using (17), (20) with setpoint $\bar{\boldsymbol{r}}_k$.
15: **end for**
16: Go to Line 1.

IV. Simulation Study

We consider an EV that travels on a single-direction two-lane road. The road includes both straight-line and curved road segments. The road coordinates are from the outer ring test track of the Japanese Automobile Research Institute proving ground in Shirosato, Japan, and the vehicle parameters used in the simulation study are obtained from a real mid-size SUV, from data-sheet, precision testbenches, and data analysis. There are surrounding vehicles maintaining either of the lanes with constant velocity. In the simulation, the obstacle set is predicted by designing lane-tracking controllers that control the OVs assuming a fixed lane over the planning horizon N_p. The desired velocity is $v_x = 20$ m/s. Hence, this is the first candidate reference velocity that the planner tries in searching for a collision-free trajectory. The gridding of the velocity setpoints is done in decrements of 2 m/s down to 10 m/s, that is, using five reference velocity setpoints. The goal region is chosen such that a path is considered to have reached the region if the endpoint is at least N_m steps long and is in the middle of either of the lanes.

TABLE I
PARAMETER VALUES USED FOR THE SIMULATION STUDY.

Parameter	value	Unit	Description
Δt	0.1	s	Sampling period vehicle dynamics
T_s	0.5	s	Sampling period in planner
N_p	20 (10)	steps (s)	Nominal planning horizon
N_m	10 (5)	steps (s)	Minimum planning horizon
N_c	5 (0.5)	steps (s)	Control horizon
n_r	36	-	# road discretization points

The planning is done in the road-aligned, local coordinate frame. However, in the simulation, the computed control inputs are used in a vehicle modeled in the global coordinate frame. Furthermore, neither the disturbance (28) nor the true motion of the vehicles are known to the planner. The disturbance is estimated online by a first-order Taylor expansion of the curvature at each time step, where the estimate and the corresponding first-order derivative are calculated from the radius of curvature, which is found from the data points by fitting a circle segment. The obstacle set (14) is determined from obstacle predictions, by using the position and velocity of each OV at the time corresponding to the beginning of each planning phase. Hence, the simulation study gives indications on the planner robustness to these uncertainties.

Table I shows the algorithm parameters. These values correspond to a weighted adjacency matrix $\boldsymbol{M} \in \mathbb{R}^{758\times758}$, out of which approximately 3100 elements are nonzero (i.e., about 0.5%). Algorithm 1 is implemented in MATLAB on a 2014 i5 laptop. We design one set of state-feedback controllers (17), (20) for the entire range $[10, 20]$ of reference velocities and construct the adjacency matrix $\boldsymbol{M}$ using the connectivity test for $v^5_{x,\mathrm{nom}} = 10$ m/s such that we ensure connectivity for the same vertices for all $v^j_{x,\mathrm{nom}} > v^5_{x,\mathrm{nom}}$. We design the edge weights as piecewise linear functions, with the lowest cost edge weight in the middle of each lane.

A. Results

Fig. 6 shows five snapshots of a situation where the EV catches up with two slower moving OVs, one in each lane. Eventually, there is no collision-free trajectory for the preferred velocity v^1_{nom}, so the planner tests the different candidate velocities in decreasing order until a solution is found. In the figure, the time at which switching between setpoints is initiated can be seen in the second and fourth subplots. The PI sets projected on the road are shown in green. When switching between different setpoints is initiated (e.g., the second plot from the left), the contraction of the invariant sets due to (25) is noticeable. The resulting trajectory in the global frame when applying the control inputs is in Fig. 7. Fig. 8 displays the velocity profile for the time period corresponding to the snapshots. The time instants when the different snapshots occur are indicated by dashed lines.

Fig. 9 shows the computation time for the planning steps across the scenario. The nominal computation time is always less than 40 ms for trajectories of at least 5 s, i.e., the ratio between real time and computing time is $\nu > 125$. The higher peaks correspond to periods when the velocity is decreased (c.f. Fig. 8). However, the computation time is always below 40 ms. The complexity grows linearly with the number of elements in the adjacency matrix and the number of obstacles [18]. Because of the fast computation, the proposed method appears suitable for use even in rapidly changing and reactive scenarios, and in automotive-grade embedded platforms whose capabilities are more limited than standard computers [17].

V. EXPERIMENTAL SETUP

For experimental validation of the method, we use the Hamster platform [43], see Fig. 10. The Hamster is a 25×20 cm mobile robot for research and prototype development. It is equipped with sensors commonly available on full-scale research vehicles, such as lidar, inertial measurement unit, GPS receiver, camera, magnetometers, and motor encoder. It uses two Raspberry PI3 for processing. The Hamster is robot operating system (ROS) compatible, hence allowing to be integrated in a ROS network. The robot uses Ackermann steering and is therefore kinematically equivalent to a full-scale vehicle, and its dynamics, such as the suspension system, resembles that of a regular vehicle. Hence, it presents itself as a suitable platform for verifying dynamic feasibility and for testing the performance of the motion planner in a realistic situation, in a safe and limited space environment.

Table II shows some of the most important algorithm parameters. The test track is a two-lane closed circuit with each lane 0.3 m wide, which implies that the lateral distance between two adjacent discretization points is 0.01 m. The longitudinal velocity is discretized equidistantly with steps of 0.04 m/s from 0.4 m/s down to 0.12 m/s, and the preferred velocity is 0.4 m/s. We design four state-feedback controllers at the velocities $\{v^1_{x,\mathrm{nom}}, \dots, v^4_{x,\mathrm{nom}}\} = \{0.4, 0.225, 0.155, 0.12\}$ m/s. The velocity level $v^i_{x,\mathrm{nom}}$ was chosen by determining the largest set for ensuring connectivity for all velocities $v_x \in [v^i_{x,\mathrm{nom}}, v^{i+1}_{x,\mathrm{nom}}]$ (c.f. Sec. III-B).

The edge weights of the connectivity graph $\mathcal{G}$ can be used as design parameters to ensure that the motion planner follows a desirable driving behavior. The final edge weights, computed as piecewise linear functions in the lateral direction, that are used for our driving experiments are shown in Fig. 11, where a heat map of the cost landscape is shown for the *nodes* of the graph, and the value associated to a node is the weight of *all* edges pointing to this node. According to Fig. 11, in the lateral dimension it is more expensive to move to a node that is located between the lanes. This strategy encourages the motion planner to find a path that stays in the center of the lanes and avoids driving between lanes. Along the time dimension, we gradually increase the edge weights, that is, it becomes more expensive to move close to a preceding OV. If the EV plans to pursue an overtaking, this will encourage the EV to switch lane as early as possible.

In the experiments, the obstacle set (14) is determined from obstacle predictions in the local frame, by using the estimated position and velocity of each OV at the time instant corresponding to the beginning of each planning phase and predicting the OVs by using a proportional mid-lane tracking controller assuming constant speed.

The number of nonzero elements of the connectivity graph $\mathcal{G}_v$ for the different longitudinal velocity levels are shown in

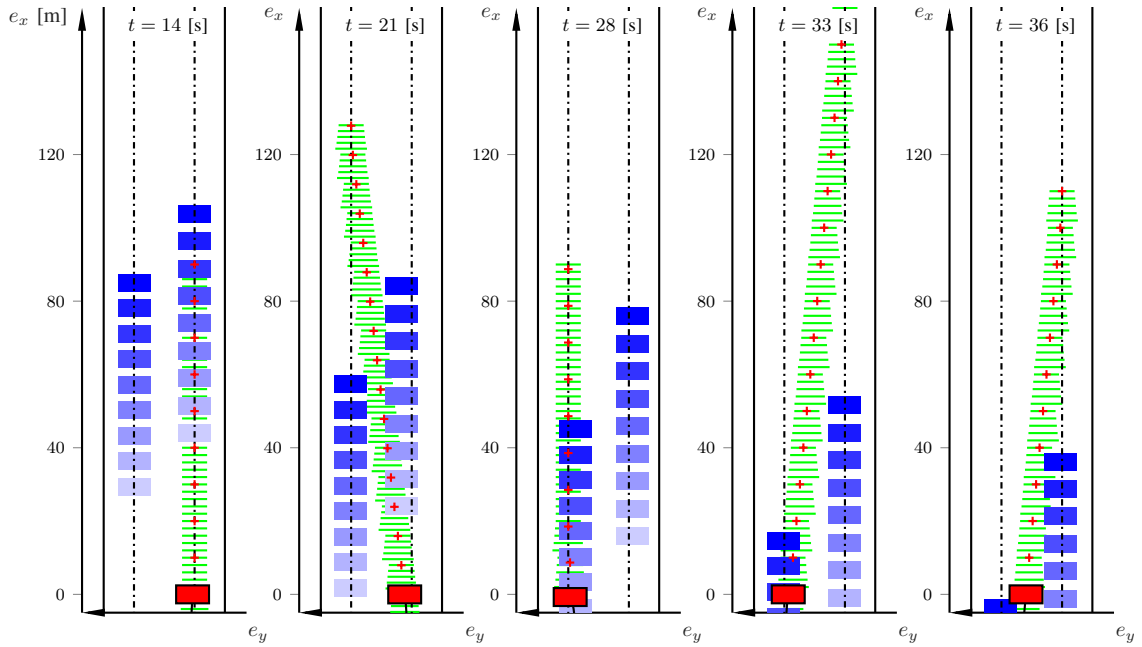

Fig. 6. Five snapshots of a situation where the EV (red) catches up with two slower moving OVs (blue), one in each lane. The desired path is indicated with red crosses and the invariant sets projected onto the road are in green. Resulting trajectory is in black and the corresponding time the EV reaches a particular point on the trajectory is enumerated to the left in each snapshot. In each snapshot, snapshots of the OVs every 0.5 s are shown in increasing color. The snapshots of the OVs correspond to the point at the time of the reference equilibrium points.

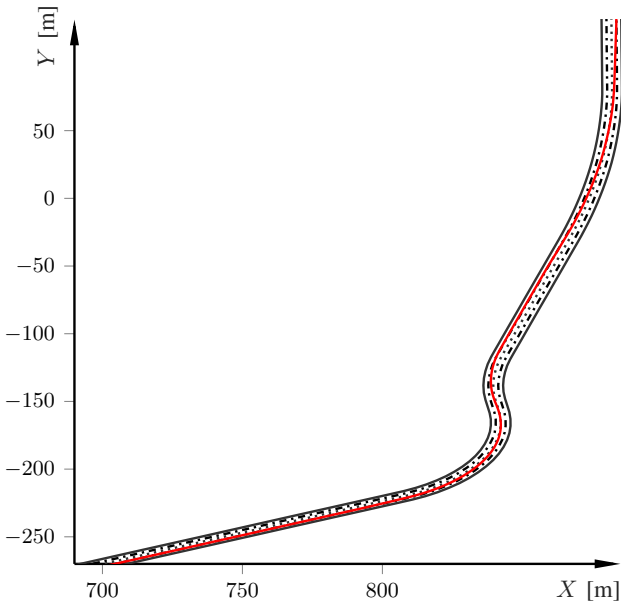

Fig. 7. The resulting trajectory in the global frame when applying the steering input resulting from Algorithm 1 over the time span in Fig. 6.

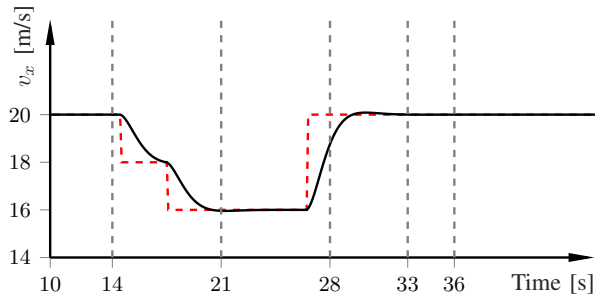

Fig. 8. The resulting velocity (black) and reference velocity setpoints (red dashed) as computed by the planner over the time span in Fig. 6. The dashed lines correspond to the snapshots in Fig. 6, numbered from left to right.

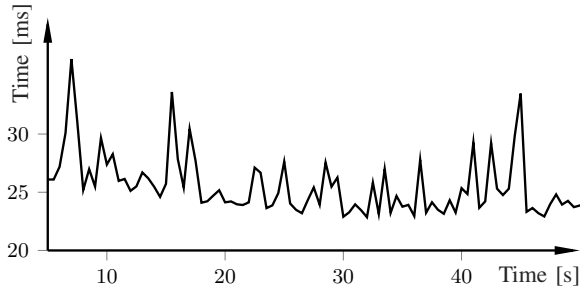

Fig. 9. Computation time over the time steps for the driving scenario (Figs. 6–7). The implementation is done in MATLAB on a 2014 i5 2.8GHz laptop.

Fig. 10. The Ackermann-steered mobile robot used in the experiments.

TABLE II
PARAMETER VALUES USED FOR THE EXPERIMENTAL EVALUATION.

Parameter	value	Unit	Description
Δt	0.1	s	Sampling period vehicle dynamics
T_s	0.5	s	Sampling period in planner
N_p	30 (15)	steps (s)	Nominal planning horizon
N_m	10 (5)	steps (s)	Minimum planning horizon
N_c	5 (0.5)	steps (s)	Control horizon
n_r	31	-	# road discretization points

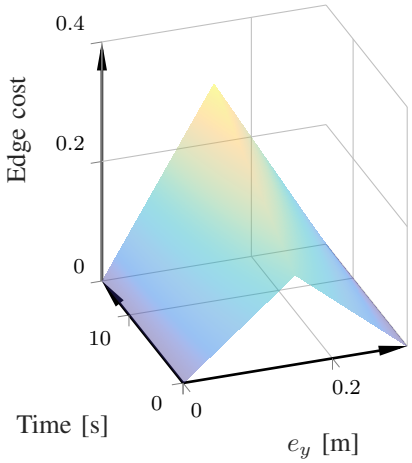

Fig. 11. Illustration of the edge weights of the connectivity graph, with the boundaries in the lateral direction being the center of each lane. It is most expensive to move to a node located between the lanes.

TABLE III
PROPERTIES OF THE CONNECTIVITY GRAPH $\mathcal{G}_v$.

v [m/s]	Nonzero elements	Sparsity [%]	Setpoint outdegree
0.4	4493	99.52	5
0.36	4493	99.52	5
0.32	4493	99.52	5
0.28	4493	99.52	5
0.24	4493	99.52	5
0.20	4493	99.52	5
0.16	2753	99.70	3
0.12	2753	99.70	3

Table III, where the velocity levels are associated with the state-feedback controller of the corresponding velocity region. The sparsity is the fraction of zero-valued elements of $\mathcal{G}_v$ and the setpoint outdegree is the maximum number of out-neighbors of any node in $\mathcal{G}_v$. The connectivity graphs for the six highest velocity are all the same. For the two lowest velocities, the connectivity graphs are the same, yet different from the former ones. In fact, for the lower velocity levels, the EV can only reach the two most adjacent setpoints in one planning step, because of the imposed input constraint set $\mathcal{U}$. Since the connectivity graphs are extremely sparse for all velocities, the graph-search problem can be computed efficiently in real-time.

For localization, we create a 2-D occupancy grid map [44]. The map is sent to a Monte-Carlo based localization system (AMCL) [45], which uses the lidar for localization in the map. The localization system has a low update frequency, does not fully utilize all the available sensing, and is furthermore subject to outliers. To account for this, we have implemented an extended Kalman filter (EKF) that estimates the vehicle state vector by fusing the position estimates from AMCL, the IMU, and odometry sensors. The EKF additionally estimates the bias of the steering-wheel sensor, and we have implemented an outlier detection scheme and filter divergence monitoring [46]. We use the lidar also for online obstacle detection.

VI. EXPERIMENTAL EVALUATION

We present results from three different scenarios, one with moving obstacles operating normally on the road and communicating their initial position and velocity, one with a static obstacle appearing suddenly in front of the EV that is only detected by lidar, and one intersection test with moving OVs detected only by lidar.

A. Overtaking of two OVs

The first scenario considers overtaking of two OVs. To show the performance of the planning module without presence of disturbances in obstacle detection, we simulate the OVs using multiple instances of an appropriately built ROS simulator, each implemented as a stand alone ROS-node, and send the obstacle information (pose and velocity) at the beginning of the planning stage to the planner.

Fig. 12 shows six snapshots for the overtaking scenario (lower). The upper plot shows the velocity setpoint (red dashed) and the estimated velocity (black). The OVs in the right and left lane have constant velocities of 0.15 m/s and 0.2 m/s, respectively. In this case, the motion planner has to reduce the reference velocity to ensure that the EV does not enter the dangerous zones around the obstacles. The EV aims to drive close to the nominal speed 0.4 m/s and the OV in the left lane is the faster among the two. Hence, the motion planner overtakes first the OV in the right lane. Once it is possible to safely plan a trajectory back to the preferred right lane, this becomes the intended motion plan.

Fig. 13 shows the tracking errors for the experiment. The tracking error is within a range of ± 1.5 cm throughout, with the tracking error below 1 cm most of the time, showing that the approximations to the model are suitable and well within the safety margins of the Lyapunov function (Remark 2). The path for the entire experiment is shown in Fig. 14.

B. Lidar-Detected Obstacle Avoidance

Here, the EV is driving on the outer lane without any OVs nearby. An OV suddenly appears in front of the EV. The OV does not send any information about position and velocity: the lidar is responsible for detecting the OV, and the position and velocity estimate is propagated to the motion planner.

Fig. 15 shows that the onboard lidar successfully detects the OV, and the motion planner can appropriately react to the sudden OV appearance. The motion planner can quickly react to the detected obstacle and re-compute a safe trajectory thanks to the fast computation time. Fig. 16 shows a snapshot of what the lidar detects. Multiple obstacles are detected, for instance, corners of some nearby objects. All detected obstacles not in the interior of the map (i.e., all but the OV) are filtered out when propagated to the motion planner. The EV switches lane to avoid the OV, and returns back once it has passed the obstacle. This behavior is very similar to the results using virtual OVs in Sec. VI-A.

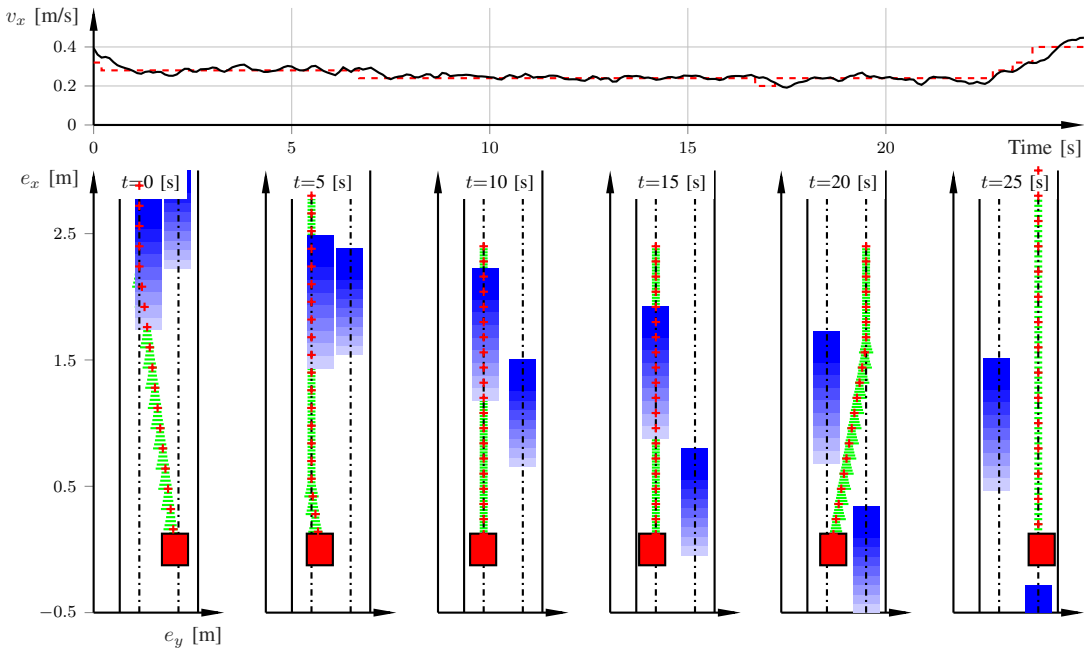

Fig. 12. Six snapshots of an experimental evaluation of overtaking of two slower moving OVs. The upper plot shows the estimated velocity (black) and reference velocity setpoints (red dashed) as computed by the motion planner. Same notation as in Fig. 6 (Sec. VI-A).

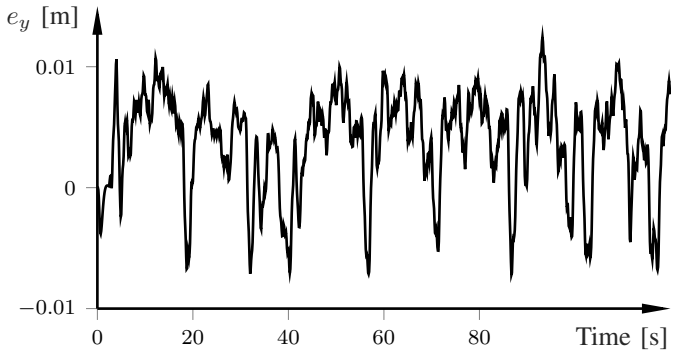

Fig. 13. Tracking error as a function of time for the same experiments as in Figure 12 (Sec. VI-A).

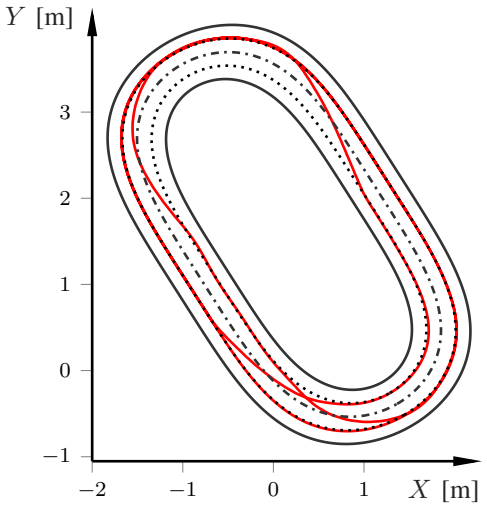

Fig. 14. The EV path for the whole experiment corresponding to Fig. 12.

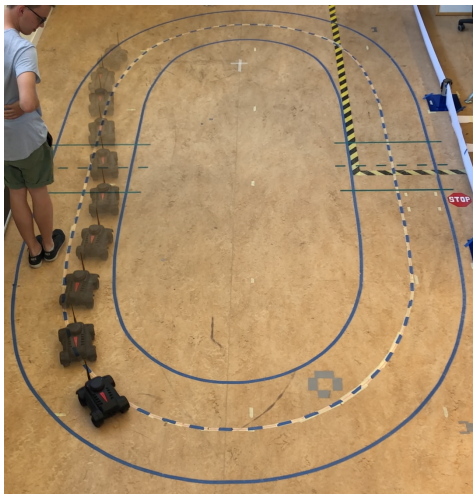

Fig. 15. Overtaking scenario with a lidar-detected static obstacle . The increased opacity illustrates the time progression.

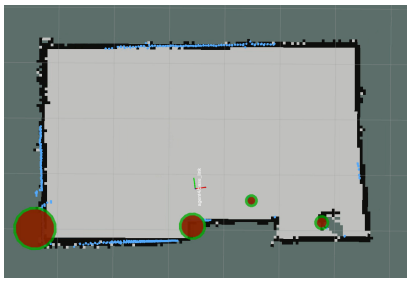

Fig. 16. Snapshot of the `rviz` window during the lidar detection experiment with one static obstacle. The small coordinate frame represents the estimated EV position, where the red axis is in the heading direction of the vehicle. The lidar-detected obstacles are illustrated by red circles with green borders. The view is rotated 90 ° anti-clockwise with respect to Figure 15.

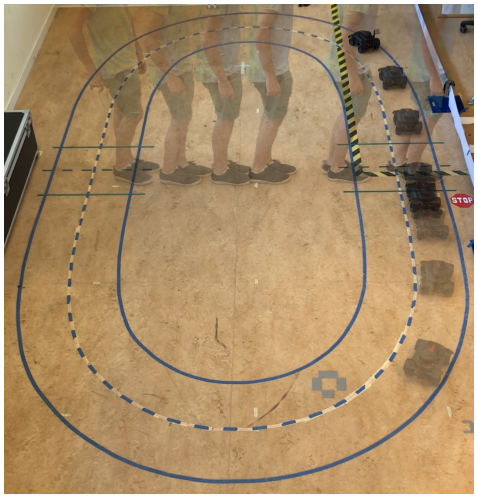

Fig. 17. Fused sequence of images from the intersection experiment with a lidar-detected moving obstacle. Increased opacity illustrates time progression.

C. Intersection with Crossing OVs

In this scenario we demonstrate how the proposed motion planner can handle intersection scenarios. We have implemented simple heuristic decision rules for signs and predicted obstacles in the EV ROI. Sign information is propagated to the motion planner. If there are any signs in the ROI, the EV enters sign mode and certain overtaking maneuvers are prevented. A decision to stop is implemented by setting the preferred velocity $v^1_{x,\mathrm{nom}} = 0$.

Fig. 17 shows the motion sequence from the intersection experiment. The EV safely stops at the stop sign and waits until the obstacle has passed, before moving on. Also in this case the OV does not provide any position or velocity information which are then estimated only from lidar sensing.

Fig. 18 shows snapshots for the intersection scenario. From the velocity plot, it is clear that the motion planner is capable of planning a smooth deceleration, stop at the stop sign, and continuing on once the intersection is clear. The prediction of the detected obstacle is seen in the snapshots for the time instants from 2.6 s to 7.8 s. Although the obstacle is actually moving along the crossing road, the predicted direction of motion is changing in time due to the noise characteristics of the lidar scans. Indeed, the estimate of the OV velocity obtained from the onboard lidar is not very reliable. Still, the detected obstacles are handled by the motion planner to produce reference trajectories that safely avoid the obstacle.

VII. Conclusion

This paper presented a method for integrating motion planning and vehicle control by exploiting PI sets. The proposed approach enforces constraints on the vehicle motion as well as avoids collisions. The method uses a graph search over lateral displacements on the road for different reference velocity setpoints, and then executes a sequence of state-feedback controllers using the nodes in the path resulting from the graph search as a sequence of corresponding target equilibria. The simulation study showed that the method can safely navigate the vehicle through tight passages where combined slow-down and lane change is needed, and the online computation requirements are modest.

The experimental study, even when using only onboard sensors for localization and obstacle detection, shows that the method is robust to sensing errors and disturbances. Furthermore, the tracking errors are within 1 cm most of the time, implying that the method indeed computes dynamically feasible, realistic trajectories. The computation for the proposed method is fast enough to react to obstacles that suddenly appear in front of the vehicle, as long as the obstacle behavior is not excessively malicious.

References

[1] D. Burton, A. Delaney, S. Newstead, D. Logan, and B. Fildes, "Evaluation of anti-lock braking systems effectiveness," RACV, Tech. Rep. 04/01, 2004.

[2] K. Reif and K.-H. Dietsche, *Bosch Automotive Handbook*, ser. Bosch Invented for life. Plochingen, Germany: Robert Bosch GmbH, 2011.

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

[4] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *Amer. Control Conf.*, Seattle, WA, May 2017.

[5] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

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

[7] K. Berntorp and S. Di Cairano, "Particle filtering for online motion planning with task specifications," in *Amer. Control Conf.*, Boston, MA, Jul. 2016.

[8] K. Berntorp, T. Hoang, and S. Di Cairano, "Motion planning of autonomous vehicles by particle filtering," *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 197–210, 2019.

[9] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.

[10] R. Allen and M. Pavone, "Toward a real-time framework for solving the kinodynamic motion planning problem," in *Int. Conf. Robot. Autom.*, Seattle, WA, May 2015.

[11] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the Urban challenge," *J. Field R*, vol. 25, no. 8, pp. 425–466, 2008.

[12] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The Stanford entry in the Urban challenge," *J. Field R.*, vol. 25, no. 9, pp. 569–597, 2008.

[13] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Veh. Syst. Dyn.*, vol. 52, no. 6, pp. 802–823, 2014.

[14] S. Di Cairano, U. Kalabić, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by MPC with guaranteed error bounds," in *Conf. Decision and Control*, Las Vegas, NV, Dec. 2016.

[15] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, 2007.

[16] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty - a control perspective," *Eur. J. Control*, vol. 24, pp. 14–32, 2015.

[17] S. Di Cairano and I. V. Kolmanovsky, "Real-time optimization and model predictive control for aerospace and automotive applications," in *Amer. Control Conf.*, Milwaukee, WI, Jun. 2018.

[18] K. Berntorp, A. Weiss, C. Danielson, I. Kolmanovsky, and S. Di Cairano, "Automated driving: Safe motion using positively invariant sets," in *Int. Conf. Intell. Transp. Syst.*, Yokohama, Japan, Oct. 2017.

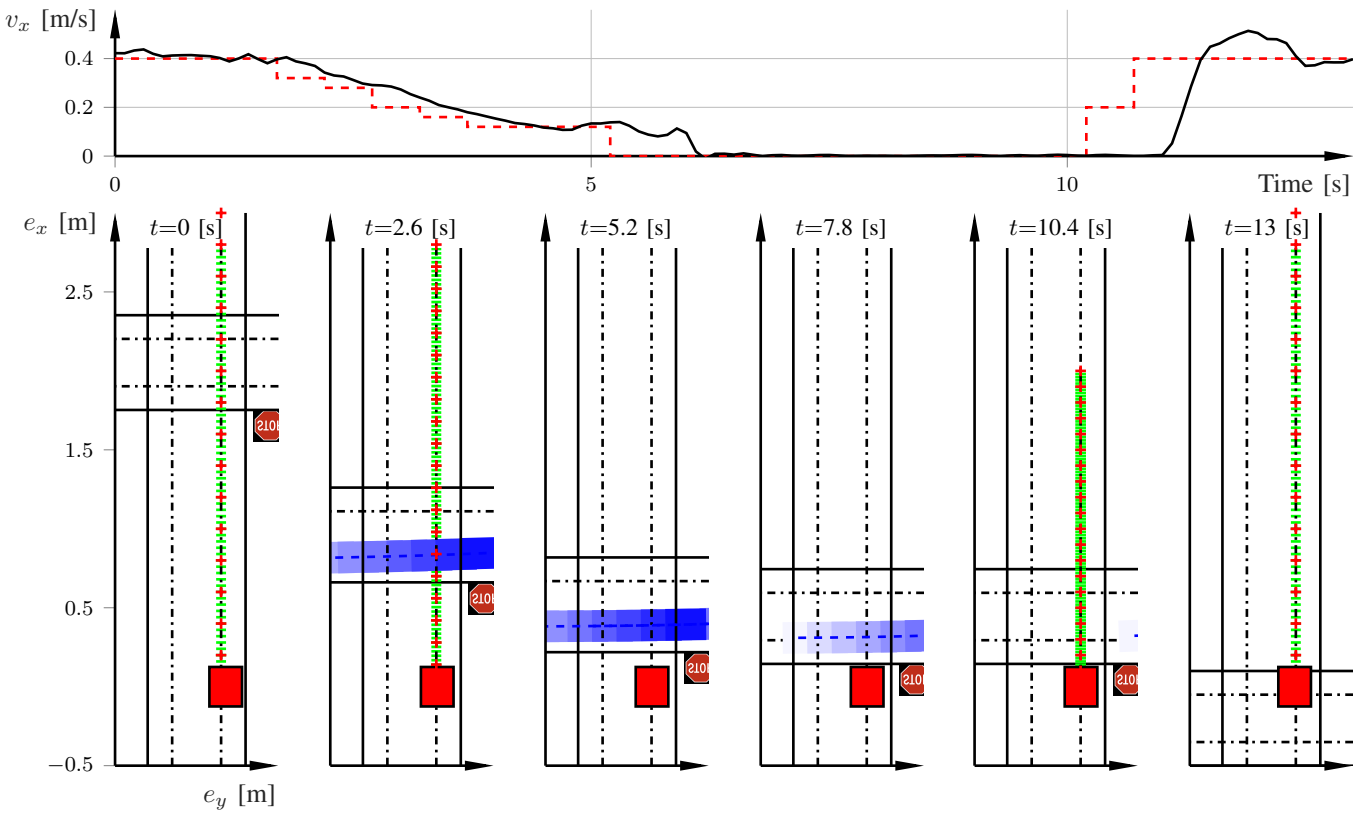

Fig. 18. Intersection scenario with a stop sign and a lidar-detected moving obstacle, corresponding to Fig. 17. Same notation as in Fig. 6.

[19] K. Berntorp, C. Danielson, A. Weiss, and S. Di Cairano, "Positive invariant sets for safe integrated vehicle motion planning and control," in *Conf. Decision and Control*, Orlando, FL, Dec. 2018.

[20] N. Murgovski and J. Sjöberg, "Predictive cruise control with autonomous overtaking," in *Conf. Decision and Control*, Osaka, Japan, 2015.

[21] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *Int. Conf. Intell. Transp. Syst.*, The Hague, Netherlands, 2013.

[22] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, "Receding horizon maneuver generation for automated highway driving," *Control Eng. Pract.*, vol. 41, pp. 124–133, 2015.

[23] S. Singh, A. Majumdar, J. J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Int. Conf. Robot. Autom.*, Singapore, May 2017.

[24] G. Franzè and W. Lucia, "The obstacle avoidance motion planning problem for autonomous vehicles: A low-demanding receding horizon control scheme," *Systems & Control Letters*, vol. 77, no. 1–10, 2015.

[25] ——, "A receding horizon control strategy for autonomous vehicles in dynamic environments," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 2, pp. 695–702, 2016.

[26] B. Schürmann, N. Kochdumper, and M. Althoff, "Reachset model predictive control for disturbed nonlinear systems," in *Conf. Decision and Control*, Maimi Beach, Fl, Dec. 2018.

[27] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *J. Robot. Res.*, vol. 36, no. 8, pp. 947–982, 2017.

[28] M. Althoff and J. M. Dolan, "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles," in *Int. Conf. Intell. Transp. Syst.*, 2011.

[29] ——, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, 2014.

[30] D. Hess, M. Althoff, and T. Sattel, "Formal verification of maneuver automata for parameterized motion primitives," in *Int. Conf. Intell. Robots and Syst.*, Chicago, Il, Sep. 2014.

[31] H. B. Pacejka, *Tire and Vehicle Dynamics*, 2nd ed. Oxford, United Kingdom: Butterworth-Heinemann, 2006.

[32] R. Rajamani, *Vehicle Dynamics and Control*. Springer-Verlag, 2006.

[33] A. Gray, M. Ali, Y. Gao, J. K. Hedrick, and F. Borrelli, "Integrated threat assessment and control design for roadway departure avoidance," in *Int. Conf. Intell. Transp. Syst.*, Anchorage, AK, 2012.

[34] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*. Dover, 2011.

[35] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 137–147, 2008.

[36] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech J.*, vol. 1, no. 1, p. 1, 2014.

[37] K. Okamoto, K. Berntorp, and S. Di Cairano, "Driver intention-based vehicle treat assessment using random forests and particle filtering," in *IFAC World Congress*, Toulouse, France, Jul. 2017.

[38] C. Danielson, A. Weiss, K. Berntorp, and S. Di Cairano, "Path planning using positive invariant sets," in *Conf. Decision and Control*, Las Vegas, NV, Dec. 2016.

[39] A. Weiss, C. Petersen, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Safe positively invariant sets for spacecraft obstacle avoidance," *J. Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 720–732, 2014.

[40] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. Philadelphia: SIAM, 1994.

[41] J.-P. Aubin, *Viability theory*. Springer Science & Business Media, 2009.

[42] S. Di Cairano, "Indirect adaptive model predictive control for linear systems with polytopic uncertainty," in *Amer. Control Conf.*, Boston, MA, Jul. 2016.

[43] Cogniteam, "The Hamster," 2018, [accessed 8-January-2018]. [Online]. Available: www.cogniteam.com/hamster5.html

[44] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, pp. 34–46, 2007.

[45] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[46] F. Gustafsson, *Statistical Sensor Fusion*. Lund, Sweden: Utbildningshuset/Studentlitteratur, 2010.

Karl Berntorp received the M.Sc. degree in Engineering Physics in 2009 and the Ph.D. degree in Automatic Control in 2014, from Lund University, Lund, Sweden. In 2008 he was a visiting researcher at Daimler AG in Sindelfingen, Germany. In 2014 he joined Mitsubishi Electric Research Laboratories in Cambridge, MA. His research is on statistical signal processing, sensor fusion, and optimization-based control, with applications to automotive, aerospace, transportation, and communication systems. His work includes design and implementation of nonlinear estimation, constrained control, and motion-planning algorithms. Dr. Berntorp is the author of more than 65 peer-reviewed international papers and patents.

Richard Bai received the M.Sc. degree in Engineering Physics from Lund University, Lund, Sweden in 2018. His research interests include motion planning and sensor fusion for control of autonomous vehicles. He is currently working as a consultant at Netlight in Stockholm, Sweden.

Karl F. Erliksson received the M.Sc. degree in Engineering Physics in 2018 from Lund University, Lund, Sweden. In 2016 he was a research fellow at Netlab at California Institute of Technology, Pasadena, CA, USA. His research interests include motion planning and control for autonomous vehicles, as well as optimization algorithms for smart grid applications. He is currently with the Advanced Analytics practice at Accenture Sweden.

Claus Danielson received his PhD in 2014 from the Model Predictive Control Laboratory at the University of California, Berkeley. He is currently a Principal Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, MA. His research interests are in predictive and constrained control. His specialty is developing methods for exploiting structure in large-scale or complex control and optimization problems. He has applied his research to a variety of fields include energy storage networks, heating ventilation and air conditioning, adaptive optics, spacecraft guidance and control, atomic force microscopy, autonomous vehicles, cancer treatment, and robotics.

Avishai Weiss is a Principal Research Scientist in the Control and Dynamical Systems group at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA. He received his Ph.D. in Aerospace Engineering in 2013 from the University of Michigan and holds B.S. and M.S. degrees from Stanford University in Electrical Engineering (2008) and Aeronautics and Astronautics (2009). His main research interests and contributions are in the areas of spacecraft orbital and attitude control, constrained control, model predictive control, and time-varying systems, in which he has authored over 45 peer-reviewed papers and patents.

Stefano Di Cairano received the Master (Laurea), and the PhD in Information Engineering in '04 and '08, respectively, from the University of Siena, Italy. He has been visiting student at the Technical University of Denmark and at the California Institute of Technology. During 2008–2011, he was with Powertrain Control R&A, Ford Research and Adv. Engineering, Dearborn, MI. Since 2011, he is with Mitsubishi Electric Research Laboratories, Cambridge, MA, where he is now the Senior Team Leader for Optimization-based Control, and a Distinguished Researcher in Control and Dynamical Systems. His research is on optimization-based control strategies for complex mechatronic systems, in automotive, factory automation, transportation systems and aerospace. His research interests include model predictive control, constrained control, particle filtering, hybrid systems, optimization.

Dr. Di Cairano has authored/co-authored more than 150 peer reviewed papers in journals and conference proceedings and 35 patents. He was the Chair of the IEEE CSS Technical Committee on Automotive Controls 2012-2015, the Chair of IEEE Standing Committee on Standards since 2016-2019, and an Associate Editor of the IEEE Transactions on Control Systems Technology.