Spatial-Based Predictive Control and Geometric Corridor Planning for Adaptive Cruise Control Coupled With Obstacle Avoidance

Mogens Graf Plessen, Daniele Bernardini, Hasan Esen, and Alberto Bemporad, *Fellow, IEEE*

***Abstract*— This paper presents an integrated control approach for autonomous driving comprising a corridor path planner that determines constraints on vehicle position, and a linear time-varying model predictive controller combining path planning and tracking in a road-aligned coordinate frame. The capabilities of the approach are illustrated in obstacle-free curved road-profile tracking, in an application coupling adaptive cruise control (ACC) with obstacle avoidance (OA), and in a typical driving maneuver on highways. The vehicle is modeled as a nonlinear dynamic bicycle model with throttle, brake pedal position, and steering angle as control inputs. Proximity measurements are assumed to be available within a given range field surrounding the vehicle. The proposed general feedback control architecture includes an estimator design for fusion of database information (maps), exteroceptive as well as proprioceptive measurements, a geometric corridor planner based on graph theory for the avoidance of multiple, potentially dynamically moving objects, and a spatial-based predictive controller. Switching rules for transitioning between four different driving modes, i.e., ACC, OA, obstacle-free road tracking (RT), and controlled braking (Brake), are discussed. The proposed method is evaluated on test cases, including curved and highway two-lane road tracks with static as well as moving obstacles.**

***Index Terms*— Adaptive cruise control (ACC), autonomous driving, corridor planning, obstacle avoidance (OA), spatial-based predictive control, vehicle dynamics control.**

I. Introduction

AUTONOMOUS driving requires handling a plethora of different driving scenarios, ranging from parking to crash avoidance on challenging road conditions. The objective is to ensure road safety and increase passenger comfort. A suitable candidate for the control strategy is model predictive control (MPC), due to its ability to incorporate system constraints and nonlinearities in a systematic way.

Rear-end collisions are frequently caused by inadequate following distances, inattentiveness on the part of the driver and unexpected, abrupt traffic standstills. Here, adaptive cruise control (ACC) can provide remedy. This is ensured by the automated adaptation of vehicle velocity and the usage of proximity sensors, such as, for example, infrared laser, radar, and video sensors. The ACC problem using MPC is tackled, for example, in [1]. In [2], an ACC method considering multiple objects by means of hybrid system theory is discussed. An ACC system trading off between tracking capability, fuel economy, and driver ride comfort can be found in [3]. The work in [4] exploits traffic signal information. In [5], a learning-based approach for autonomous car following on a highway is proposed. All the aforementioned references focus on longitudinal dynamics only, not considering steering maneuvers for obstacle avoidance (OA).

OA by active steering (AS) becomes necessary to avoid a crash with a (moving) object in front, if braking alone is not sufficient or undesirable. In general, avoidance of one or more obstacles results in a nonconvex optimization problem, as the set of possible safe trajectories around the obstacle is nonconvex (i.e., passing it from left or right). Typically, hierarchical two-level controllers are employed, featuring a high-level path planner and a low-level path tracking controller. In [6], at the top level, a trajectory avoiding an obstacle is computed based on a simple point-mass vehicle model. At the bottom level, an MPC controller using a higher fidelity four wheel model is employed to track this trajectory. Although this decomposition enabled real-time implementation, the trajectories generated by the point-mass path planner were not always feasible [7]. In the latter reference, on the top level, a nonlinear MPC problem is solved using a nonlinear bicycle vehicle model described in a road-aligned coordinate system to generate an obstacle-free path. In the bottom level, the spatial trajectory is transformed back to a time-dependent trajectory by coordinate transformation. A second nonlinear MPC problem is then solved using a higher fidelity four wheel model. Thus, two different vehicle models are employed and two different nonlinear MPC problems (one in the space domain and one in the time domain) are solved at every sampling time. A road-aligned coordinate frame is also employed in [8] for a driver assist system. In [9], a one-level approach for OA of two static obstacles is presented. A transformation to a spatial-dependent coordinate system is particularly useful in lane-keeping applications, since the centerline position reference is zero. To avoid the computational burden involved by real-time trajectory generation using a complex dynamical vehicle model, an offline-generated library of motion primitives is used within a hierarchical control framework for the high-level path planning task [10], [11].

Manuscript received March 9, 2016; revised November 23, 2016; accepted January 24, 2017. Date of publication February 17, 2017; date of current version December 14, 2017. Manuscript received in final form January 31, 2017. Recommended by Associate Editor M. Tanelli.

M. Graf Plessen and A. Bemporad are with the IMT Institute for Advanced Studies Lucca, 55100 Lucca, Italy (e-mail: mogens.plessen@imtlucca.it; alberto.bemporadg@imtlucca.it).

D. Bernardini is with ODYS Srl, 55100 Lucca, Italy (e-mail: daniele.bernardini@odys.it).

H. Esen is with DENSO AUTOMOTIVE Deutschland GmbH, 85386 Eching, Germany (e-mail: h.esen@denso-auto.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2017.2664722

The main objective of the control strategy presented in this paper is to perform ACC coupled with AS for OA on arbitrarily curved road tracks. We use the road-aligned coordinate frame, handle system nonlinearities by linearization and discretization around state and input reference trajectories, and formulate a spatial-based MPC problem in the form of a quadratic program (QP). The state-transition matrices are *space*-varying. For simplicity, in the following, we still maintain the expression linear time-varying MPC (LTV-MPC), even though the prediction horizon is defined in terms of space, not time. The contribution includes further handling of noisy/failing sensors, the discussion of switching rules for four different driving modes, and a geometric corridor planning algorithm for the avoidance of multiple static and dynamic obstacles. The piecewise-affine (PWA) reference trajectories returned by the path planner are either spontaneously smoothed by the spatial-based predictive controller, or first processed according to [12]. This paper focuses exclusively on autonomous driving. In contrast, our companion paper [13] treats coordinated driving of multiple automated vehicles, whereby every automated vehicle is assumed to be equipped with the control architecture presented here.

This paper is organized as follows. Section II describes the vehicle dynamics. Autonomous ACC, OA, obstacle-free road tracking, and controlled braking are discussed in Section III. Numerical simulation results are presented in Section IV, before concluding with Section V.

A. Notation

We adopt the notation $\dot{x} = dx/dt$ for time derivatives and omit the time dependence of variable x for brevity. Furthermore, we use $x' = dx/ds$ to denote spatial derivatives. We denote $\|x\|_Q^2 \triangleq x^T Q x$ for $x \in \mathbb{R}^n$ and a positive definite matrix $Q \in \mathbb{S}_{++}^n$.

II. Vehicle Dynamics and Road-Profile Modeling

This section states the nonlinear vehicle model and dynamical description for the longitudinal distance between an automated vehicle and a leading object. First, the time-dependent system description is given, from which its spatial-based representation is derived.

A. Time-Dependent Modeling

We consider a nonlinear vehicle model, the so-called "bicycle model," involving eight states, $z^t = [x,\ y,\ \psi,\ v_x,\ v_y,\ \omega,\ \omega_f,\ \omega_r]^T$, and four control inputs, $u^t = [u_t,\ u_b,\ u_\delta,\ u_g]^T$, whereby x and y indicate the center of gravity (CoG) in the inertial frame, ψ denotes the yaw angle (relative to the inertial frame), the longitudinal and lateral velocities are given by v_x and v_y, and where ω, ω_f, and ω_r are yaw rate, front, and rear wheel speed, respectively. The inputs u_t and u_b denote throttle and brake pedal positions. The front steering input is u_δ and the integer transmission gear control variable is denoted by $u_g \in \{1,\ 2,\ 3,\ 4\}$. Thus, dropping the arguments and dependencies for brevity, the time-dependent vehicle model

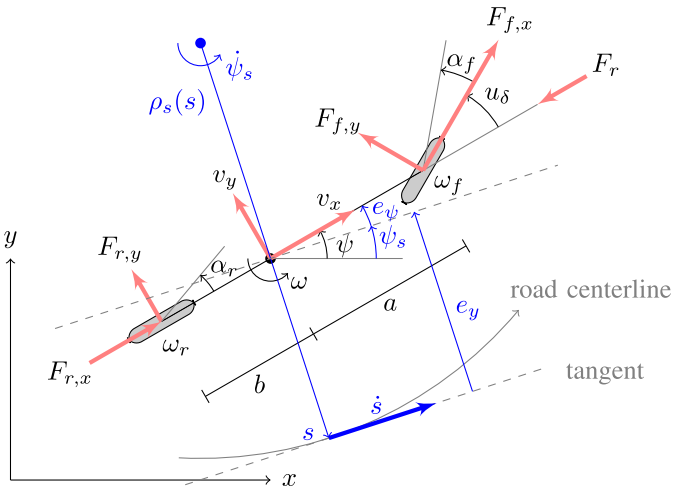

Fig. 1. Nonlinear dynamic bicycle model including the representation of the curvilinear coordinate system.

as provided by DENSO is

$$\dot{x} = v_x \cos(\psi) - v_y \sin(\psi) \tag{1a}$$
$$\dot{y} = v_x \sin(\psi) + v_y \cos(\psi) \tag{1b}$$
$$\dot{\psi} = \omega, \tag{1c}$$
$$\dot{v}_x = \frac{1}{m}(F_{f,x}\cos(u_\delta) + F_{r,x} - F_{f,y}\sin(u_\delta) - F_r) + \omega v_y \tag{1d}$$
$$\dot{v}_y = \frac{1}{m}(F_{f,x}\sin(u_\delta) + F_{f,y}\cos(u_\delta) + F_{r,y}) - \omega v_x \tag{1e}$$
$$\dot{\omega} = \frac{1}{I_z}(a(F_{f,x}\sin(u_\delta) + F_{f,y}\cos(u_\delta)) - bF_{r,y}) \tag{1f}$$
$$\dot{\omega}_f = \frac{1}{I_\omega}(T_d - T_b - r_t F_{f,x}) \tag{1g}$$
$$\dot{\omega}_r = \frac{1}{I_\omega}(T_d - T_b - r_t F_{r,x}) \tag{1h}$$

where $F_{f,x}(v_x, u_\delta, v_y, \omega, \omega_f)$ denotes the longitudinal force on the front tire. It is a nonsmooth nonlinear function, modeled here as a lookup table calibrated experimentally. $F_{r,x}(v_x, \omega_r)$ is the force acting longitudinally on the rear wheel, which is also modeled by a lookup table. The lateral forces on the front and rear wheels are denoted by $F_{f,y}(u_\delta, v_y, \omega, v_x)$ and $F_{r,y}(v_x, v_y, \omega)$, respectively. The engine is modeled by a 2-D lookup table that maps engine speed and throttle input to engine torque. The drive torque $T_d(u_t, \omega_f, u_g)$ is ultimately computed from engine torque via an algebraic relation. The brake torque $T_b(u_b)$ is modeled as a linear function of the brake input signal. The vehicle mass, yaw, and wheel inertia are indicated by m, I_z, and I_ω, respectively. The quantities r_t, a, and b denote the tire radius, distance of CoG from front, and rear axle, respectively. We compactly summarize the system model (1) as $\dot{z}^t = f^t(z^t, u^t)$. It is further shown in Fig. 1. Note that, although the proposed dynamically model contains several simplifications, it will turn out to be adequate for MPC design.

For ACC, we introduce a simple discrete-time difference equation to model the longitudinal distance between the

controlled vehicle and a leading object, that is

$$d_{l+1}^{\text{obj}} = d_l^{\text{obj}} + T_s\left(v_l^{\text{obj}} - v_{x,l}\right) \tag{2}$$

where v_l^{obj} is the object velocity at time $t = lT_s$, $l > 0$, and the sampling time T_s is potentially dependent on velocity $v_{x,l}$.

B. Spatial-Based Modeling

In the following, the transformation of the time-dependent vehicle model (1) into a spatial-based model is presented. We consider a curvilinear or road-aligned coordinate system (see Fig. 1). The quantity e_ψ represents the difference between the yaw angle ψ and the centerline heading ψ_s. The lateral displacement of the vehicle CoG with respect to the centerline is e_y. The derivation of the spatial-based vehicle model follows [7]. We introduce the variable s, representing the distance along the centerline, and aim at modeling the dynamics of e_ψ and e_y as the functions of space s (instead of time t). The variables e_ψ and e_y will replace x, y, and ψ in the state vector of the model. We define ρ_s as the radius of curvature, and v_s as the projected vehicle speed along the direction of the centerline. Then, the following kinematic equations can be deduced from Fig. 1:

$$\begin{aligned} v_s(s) &= rho_s(s) - e_y(s))\dot{\psi_s}(s) \\ v_s(s) &= v_x(s)\cos(e_\psi(s)) - v_y(s)\sin(e_\psi(s)). \end{aligned}$$

The vehicle velocity $\dot{s}$ along the path is thus obtained as $\dot{s} = \rho_s(s)\dot{\psi_s}(s) = \rho_s(s)v_s(s)/(\rho_s(s) - e_y(s)) = \rho_s(s)(v_x(s)\cos(e_\psi(s)) - v_y(s)\sin(e_\psi(s)))/(\rho_s(s) - e_y(s))$, and the time derivatives of the heading error e_ψ and of the displacement error e_y can be defined as

$$\begin{aligned} \dot{e}_\psi(s) &= \dot{\psi}(s) - \dot{\psi_s}(s) \\ \dot{e}_y(s) &= v_x(s)\sin(e_\psi(s)) + v_y(s)\cos(e_\psi(s)). \end{aligned}$$

We here assume $\dot{s} \neq 0$ (a remark follows in Section II-E) at any time and observe that the spatial derivative can be expressed as a function of the time derivative, namely, $d(\cdot)/ds = d(\cdot)/dtdt/ds = d(\cdot)/dt1/\dot{s}$. Then, the spatial derivatives for all the quantities of interest are $v_x' = \dot{v}_x/\dot{s}$, $v_y' = \dot{v}_y/\dot{s}$, $\psi' = \dot{\psi}/\dot{s}$, $e_\psi' = \dot{\psi}/\dot{s} - \psi_s'$ and $e_y' = \dot{e}_y/\dot{s}$. As an example, the spatial derivative of velocity v_x is given by $v_x' = ((1/(m))(F_{f,x}\cos(u_\delta) + F_{r,x} - F_{f,y}\sin(u_\delta) - F_r) + \omega v_y/(\rho_s(v_x\cos(e_\psi) - v_y\sin(e_\psi))/(\rho_s - e_y)))$. Let the gear variable u_g be defined by means of an external logic, that, for simplicity, we assume is velocity-dependent, so to treat u_g as a given (varying) parameter. The complete spatial-dependent vehicle model is then summarized by

$$z' = f^s(z, u) \tag{3}$$

with states $z = [v_x,\ v_y,\ \omega,\ e_\psi,\ e_y,\ \omega_f,\ \omega_r]^T$ and controls $u = [u_t,\ u_b,\ u_\delta]^T$, and where all variables are now a function of the space parameter s along the lane centerline.

Employing the relation $T_s = (D_s/(\dot{s}_k))$ between the sampling time T_s (in seconds) and the sampling space D_s (in meters), where $\dot{s}_k$ is the velocity of our vehicle at position $s = kD_s$ along the lane centerline with positive integer k, we obtain

$$T_s = D_s\frac{(\rho_k - e_{y,k})}{\rho_{s,k}(v_{x,k}\cos(e_{\psi,k}) - v_{y,k}\sin(e_{\psi,k}))}. \tag{4}$$

Thus, we can convert the distance dynamics (2) to the space domain by substitution, obtaining

$$d_{k+1}^{\text{obj}} = d_k^{\text{obj}} + \frac{D_s(\rho_k - e_{y,k})\left(v_k^{\text{obj}} - v_{x,k}\right)}{\rho_{s,k}(v_{x,k}\cos(e_{\psi,k}) - v_{y,k}\sin(e_{\psi,k}))}. \tag{5}$$

Notice that, in contrast to the time domain, the dynamics of the longitudinal distance in (5) are nonlinear in the space domain. Furthermore, we point out that given a starting distance with respect to a leading object, we can write out the recursion of (5) as just a function of the automated vehicle's states.

C. Linearization and Discretization

For the formulation of a computationally tractable QP related to the MPC design, we linearize (3) and (5) using a first-order Taylor approximation around reference trajectories z^{ref} and u^{ref}, and then apply exact discretization using [14]. As a result, we obtain the discrete-time model

$$\begin{aligned} z_{k+1} &= A_k z_k + B_k u_k + g_k \\ d_{k+1}^{\text{obj}} &= d_k^{\text{obj}} + a_{d,k} z_k + g_{d,k} \end{aligned}$$

where $A_k \in \mathbb{R}^{7\times7}$, $B_k \in \mathbb{R}^{7\times3}$, $g_k \in \mathbb{R}^{7\times1}$, $d_k^{\text{obj}} \in \mathbb{R}$, $a_{d,k} \in \mathbb{R}^{1\times7}$, $g_{d,k} \in \mathbb{R}$, and $k \in \mathbb{N}$ indexes steps over distance $s \in [kD_s, (k+1)D_s]$.

As mentioned earlier, the time-dependent and spatially dependent nonlinear vehicle dynamics involve lookup tables. This implies that there are not purely analytical expressions for some of the partial derivatives. Methods for the computation of partial derivatives of functions involving lookup tables are finite differences, chain-rule-based techniques still employing the lookup tables, and fitting parametric algebraic functions before computing partial derivatives via the chain rule. In general, the "differentiation of noise" has to be avoided by smoothing (noisy) data lookup tables. In addition to being more robust, the parametric fitting method is usually computationally fastest. Therefore, it is here employed using a polynomial and exponential fit to the 2-D and 1-D lookup table data, respectively.

D. Road Curvature Modeling

We assume that a finite number of concatenated road GPS coordinates is available, $x_{s,i}$, $y_{s,i}$ for $i = 0, 1, \ldots, N^{\text{road}}$. Then, the corresponding distance along the road is $d_{s,i} = d_{s,i-1} + ((x_{s,i} - x_{s,i-1})^2 + (y_{s,i} - y_{s,i-1})^2)^{1/2}$, initialized with some $d_{s,0}$. In order to treat the distance along the road as the dependent parameter s, we either set $s_i = d_{s,i}$ or interpolate to any arbitrary grid (e.g., uniformly spaced). The radius of curvature $\rho_s(s)$ at position s along the road centerline is computed analytically as $\rho_s(s) = (((x_s(s)')^2 + (y_s(s)')^2)^{3/2}/((y_s(s)''x_s(s)' - x_s(s)''y_s(s))))$. By forward finite differences, we can approximate $x_s(s)' \approx (x_s(x+h) - x_s(s)/(h))$, $x_s(s)'' \approx (x_s(s+h) - 2x_s(s) + x_s(s-h)/(h^2))$,

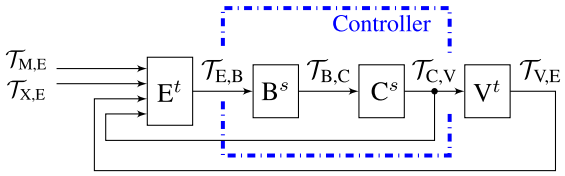

Fig. 2. Closed-loop feedback control architecture for an automated vehicle. E^t (time-domain estimator): sensor fusion/model-based state estimation, adaptive driver model, and perception/surrounding modeling. B^s (spatial-based builder): conversion from time to space domain, corridor planning, reference signal, and constraints generation as well as driving mode selection. C^s (spatial-based controller): online LTV-MPC formulation and QP-solver. V^t (time-based vehicle model): low-level controllers and nonlinear vehicle dynamics. $\mathcal{T}_{M,E}$: maps and offline-generated database. $\mathcal{T}_{X,E}$: exteroceptive measurements, i.e., surrounding perception. $\mathcal{T}_{V,E}$: proprioceptive measurements. $\mathcal{T}_{E,B}$: perception and localization information. $\mathcal{T}_{B,C}$: constraint, reference, and mode selection information. $\mathcal{T}_{C,V}$: high-level controls.

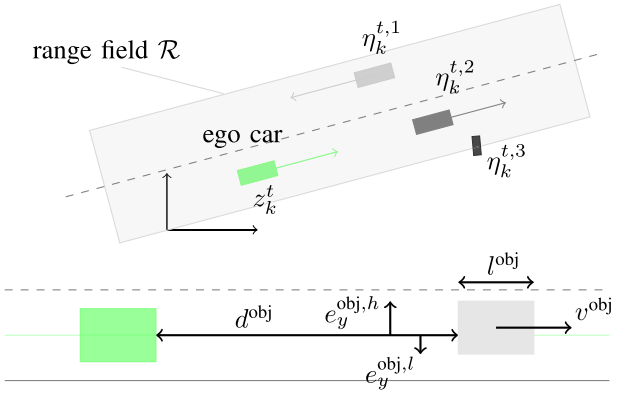

Fig. 3. Top: measurement model (proprioceptive and exteroceptive). Within a particular "range field" $\mathcal{R}$ surrounding the ego car, objects are assumed to be detectable. Bottom: illustration of the exteroceptive measurement model with five states. The red and gray areas denote the ego and a leading car, respectively.

where h is the step size. We further approximate $\psi_s(s) \approx \arctan((y_s(s+h) - y_s(s)/x_s(s+h) - x_s(s))))$ and $\psi_s(s)' \approx (\psi_s(s+h) - \psi_s(s)/(h))$. Finally, for numerical stability, we saturate $\rho_s(s) = 10^8$ (this corresponds to a drift of 5 mm on 1-km straight road), if $\rho_s(s) \geq 10^8$, to account for straight roads where $\rho_s(s) \to \infty$.

E. Extension to the Whole Operating Range

As outlined earlier, a singularity for $v = (v_x^2 + v_y^2)^{1/2} = 0$ is characteristic for the *dynamic* spatial-based system description. Let us now consider a *kinematic* vehicle model lacking de/acceleration dynamics. The most popular nonlinear kinematic bicycle model is $[\dot{x},\ \dot{y},\ \dot{\psi}]^T = [v\cos(\psi), v\sin(\psi), v/l\tan(u_\delta)]^T$, where l denotes the wheelbase [15]. Following the procedure from Section II-B, we derive the spatial-based equivalence as $[e'_\psi,\ e'_y]^T = [(\rho_s - e_y/(\rho_s l\cos(e_\psi)))\tan(u_\delta) - \psi'_s,\ (\rho_s - e_y/(\rho_s\tan(e_\psi)))]^T$, which is now entirely *velocity-independent*. This is relevant, since the whole operating range (speed) of automated vehicles can now be served. Kinematic time-dependent vehicle models exhibit a linear dependence on velocity v. This dependence is always eliminated through the transformation to a spatial-based coordinate system. Consider the transformation $(d(\cdot)/(ds)) = (d(\cdot)/(dt))(1/(\dot{s}))$ (see Section II-B) and further that $\dot{s}$ is proportional to v for kinematic vehicle models. Thus, employing one dynamic and one kinematic vehicle model for higher speeds and velocities close to 0, respectively, permits control design entirely spatial-based with state transitions being space- instead of time-varying.

III. Adaptive Cruise Control Coupled With Obstacle Avoidance

In this section, we discuss the control architecture of Fig. 2, with which an automated vehicle, referred to as the ego car, may be equipped. For a platoon of vehicles [16], every one of them may be equipped with it.

A. State Estimation and Environment Modeling

We assume there exists an area surrounding the ego car, referred to as "range field" $\mathcal{R}$ with length $l_\mathcal{R}$ at front, and in which objects are detectable (see Fig. 3). Only for proof of concept it is modeled as rectangular. The range field may be time-varying. It is typically realized through Lidar systems [17], [18]. In addition, *car-2-car* communication systems may be used, for example, for sensor redundancy. However, more importantly, they can extend the visibility of otherwise shielded vehicles, and be employed for higher-level services [19] and vehicle coordination tasks [13].

We define as proprioceptive measurements ($\mathcal{T}_{V,E}$) all of a subset of the vehicle state vector $z^t \in \mathbb{R}^8$. As exteroceptive measurements ($\mathcal{T}_{X,E}$), we define any static or mobile objects within the range field, denoting the information retrieved about the objects at sampling time kT_s by $\eta_k^{t,i}$, $\forall i = 1,\ldots,N^{\mathrm{obj}}$, whereby N^{obj} is the number of identified objects. We set

$$\eta_k^{t,i} = \left[d_k^{\mathrm{obj},i},\ e_{y,k}^{\mathrm{obj},h,i},\ e_{y,k}^{\mathrm{obj},l,i},\ l_k^{\mathrm{obj},i},\ v_{x,k}^{\mathrm{obj},i}\right]^T \tag{6}$$

fitting all objects as rectangles with particular length and width aligned with the road at the road-projected object position (see Fig. 3). This environment modeling approach is tailored to our control design. Vector $\eta_k^{t,i}$ represents the starting state from which the movement of the object can be predicted over a spatial horizon. The object lateral displacements with respect to road centerline, $e_{y,k}^{\mathrm{obj},l,i}$ and $e_{y,k}^{\mathrm{obj},h,i}$, and the object length $l_k^{\mathrm{obj},i}$ allow to formulate constraints on e_y. As the database information ($\mathcal{T}_{M,E}$), we assume the following road information vector $r_i = [x_{s,i}, y_{s,i}, \psi_{s,i}, \psi'_{s,i}, \rho_{s,i}, v_{x,i}^{\mathrm{road}}, e_{y,i}^{\mathrm{road,max}}, e_{y,i}^{\mathrm{road,min}}, s_i]$ to be available along the road centerline at samples $i = 0, 1, \ldots, N^{\mathrm{road}}$, where $v_{x,i}^{\mathrm{road}}$ denotes the speed limits, and $e_{y,i}^{\mathrm{road,max}}$ and $e_{y,i}^{\mathrm{road,min}}$ indicate the lane width along the track.

Model-based recursive estimation techniques are required for sensor fusion and reconstruction of system states in the presence of model uncertainty and unmeasured states. A nonlinear recursive filtering approach is the extended Kalman filter (EKF) (see [20]–[22]). It is often suitable for approximately Gaussian, i.e., unimodal noise. Due to the nonlinear vehicle dynamics, we employ a *discrete* EKF (dEKF) to compute an estimate $\hat{z}_k^t$ of z_k^t. Note that for the estimator design, we linearize and discretize using the *time* dynamics [see (1)].

This is in contrast to the control design, where the spatial-based system (3) is employed. For exteroceptive measurements, we similarly design one Kalman filter per object, $i = 1, \ldots, N^{\text{obj}}$, to obtain $\hat{\eta}_k^{t,i}$. To deal with situations in which detection of an object is temporarily lost, e.g., due to turning or shadow cones, we ensure that an object estimate is still returned using now only the prior update of the Kalman filter equations (not the model-correcting measurement update). Only after a certain number of consecutive steps without new measurement, the object is dismissed and the Kalman filter reinitialized. Throughout the remainder of this paper, the dEKFs are *always* active, i.e., they process any obtained measurements (even if noise-free and all states are measured).

B. Conversion From Time to Space Domain

The time-based estimate $\hat{z}_k^t \in \mathbb{R}^8$ returned by the dEKF is converted to the spatial-based $\hat{z}_k \in \mathbb{R}^7$ and $\hat{s}_k$ by using the road information. All fixed-body estimates of vehicle states $v_{x,k}$, $v_{y,k}$, ω_k, $\omega_{f,k}$, and $\omega_{r,k}$ are the same in both domains. For computation of $\hat{e}_{\psi,k}$ and $\hat{e}_{y,k}$, we first project the current ego car position to the road centerline (projection of a point mass to a PWA line). Then, we determine $\hat{e}_{\psi,k} = \hat{\psi}_k^t - \hat{\psi}_s^{\text{proj}}$ and $\hat{e}_{y,k} = c((\hat{x}_k^t - \hat{x}_s^{\text{proj}})^2 + (\hat{y}_k^t - \hat{y}_s^{\text{proj}})^2)^{1/2}$, where $\hat{x}_s^{\text{proj}}$, $\hat{y}_s^{\text{proj}}$, and $\hat{\psi}_s^{\text{proj}}$ indicate the position and orientation of the road centerline at the vehicle's projection point, and where the sign of $\hat{e}_{y,k}$ is specified by $\theta = \tan^{-1}((\hat{y}_k^t - \hat{y}_s^{\text{proj}})/(\hat{x}_k^t - \hat{x}_s^{\text{proj}})))$, and $c = -1$ if $\theta - \hat{\psi}_s^{\text{proj}} \in (0, -\pi]$, or $c = 1$ otherwise.

C. Driving Mode Selection Heuristics

The objectives of ACC (with distance-keeping capabilities) and automated OA (approaching and overtaking of a leading object) are by definition conflicting. We distinguish between four driving modes: ACC, OA, object-free road tracking (RT), and controlled braking (Brake).

For ACC-mode activation, we make use of knowledge about the ego car's braking capabilities. We assume that, for an ego car with velocity $v_{x,0}$ at position s_0 and for a given road surface and profile, a maximal possible brake deceleration $a_b^{\max}$ can be determined. In case of a constant deceleration, the position of the car after a time interval t can be computed as $s_t = a_b^{\max}(t^2/(2)) + v_{x,0}t + s_0$. We further assume a leading object with constant velocity v_0^{obj} and starting position s_0^{obj}, so that its position after time t is $s_t^{\text{obj}} = s_0^{\text{obj}} + v_0^{\text{obj}}t$. For crash avoidance in the interval $[0, t]$, it has to hold $s_t < s_t^{\text{obj}} - d_{\min}$, where $d_{\min}$ is a safety minimum distance. The time interval of interest is defined by N, the number of prediction steps N, i.e., $t = NT_s$. Defining $d_0^{\text{obj}} = s_0^{\text{obj}} - s_0$, we can compute the crash-critical initial velocity of the ego car as $v_{x,0} = (1/(NT_s))(d_0^{\text{obj}} + v_0^{\text{obj}}NT_s - (a_b^{\max}/(2))(NT_s)^2 - d_{\min})$, resulting in a final position $s_{NT_s} = s_{NT_s}^{\text{obj}} - d_{\min}$. Introducing an arbitrary safety factor $c \in (0, 1)$ to trigger steering at a lower velocity, the criterion for switching to OA mode is thus to check at time kT_s if $\hat{v}_{x,k} > \hat{v}_{x,k}^{\text{crit},i^*}$ holds, with $\hat{v}_{x,k}^{\text{crit},i^*} = (1/(NT_s))(c\hat{d}_k^{\text{obj},i^*} + \hat{v}_{x,k}^{\text{obj},i^*}NT_s - (a_b^{\max}/(2))(NT_s)^2 - d_{\min})$, where $a_b^{\max}$ is a function of $\hat{v}_{x,k}$ and $\hat{s}_k$. The critical velocity can be refined by considering detailed knowledge about braking dynamics, model-based probabilistic predictions of leading vehicle trajectories, or intervehicular communicated braking trajectories. In all cases, $\hat{v}_{x,k}^{\text{crit},i^*}$ is determined from solving $s_{NT_s} = s_{NT_s}^{\text{obj}} - d_{\min}$.

Algorithm 1 Driving Mode Selection (ACC, OA, RT, and Brake)

Input: range field $\mathcal{R}$, velocity- and safety margin-adjusted location information of all N^{obj} obstacles in $\mathcal{R}$.
Order objects according to increasing s-position along the corridor and identify corresponding road bounds.
Dismiss objects significantly faster and in front or slower and behind the ego car position.
Group objects longitudinally and laterally overlapping.
Find an object of interest $i^\star \in \{1, \ldots, N^{\text{obj}}\}$ with non-zero velocity (usually the closest to the ego car) for ACC.
If there exists a neighboring lane which allows an obstacle avoidance maneuver:
    If $(\hat{d}_k^{\text{obj},i^*} > d_{\min})$ && $(\hat{v}_{x,k}^{\text{obj},i^*} \leq v_{x,k}^{\text{road}})$ && $(\hat{v}_{x,k}^{\text{obj},i^*} > 0)$ && $(\hat{v}_{x,k} < \hat{v}_{x,k}^{\text{crit},i^*})$ && $(\hat{v}_{x,k}^{\text{obj},i^*} > v_{x,k}^{\text{obj,min}})$: ACC;
    Elseif there are objects within $\mathcal{R}$: OA;
    Elseif there are no objects left after Step 3: RT **End**.
Else
    If $(\hat{d}_k^{\text{obj},i^*} > d_{\min})$ && $(\hat{v}_{x,k}^{\text{obj},i^*} \leq v_{x,k}^{\text{road}})$ && $(\hat{v}_{x,k}^{\text{obj},i^*} > 0)$ && $(\hat{v}_{x,k} < \hat{v}_{x,k}^{\text{crit},i^*})$: ACC;
    Elseif there still are objects within $\mathcal{R}$: Brake;
    Elseif there are no objects left after Step 3: RT **End**.
End If

A second consideration is to overtake a leading object whose velocity is too far off the road reference velocity. Naturally, this is allowable only in case of an available object-free neighboring lane permitting an OA maneuver. A possible heuristic for the minimal object velocity is $v_{x,k}^{\text{obj,min}} = v_{x,k}^{\text{road}}(0.4 + \max(0, (v_{x,k}^{\text{road}} - 30/(130 - 300.5))))$.

A basic driving mode selection logic is described by Algorithm 1. Its implication on reference trajectories, constraints, and weights are disussed in Sections III-D to III.F. We remark that the ACC mode may be active while simultaneously performing OA, of, e.g., static or quasi-static (very slow) objects. A neighboring lane may not only be occupied due to an advancing vehicle but also because of a car approaching quickly from behind (a typical highway driving situation). Ultimately, an alternative triggering method may be based on a slack variable exceeding the minimum distance constraint. However, this is guaranteed to always trigger one sampling time later than object detection, since an MPC problem has to be solved first to output the slack variable. In simulations, this short delay could already render a successful OA maneuver impossible, even though it was feasible using the method described before (acting directly upon first object detection).

D. Spatial-Based Predictive Control

The control commands are applied for a duration of T_s time units before new measurements are processed and the controls are updated. Synchronization between the sampling interval and the spatial prediction step is needed. Any arbitrarily spaced discretization grid may be chosen. For a uniformly spaced

prediction horizon, we relate D_s and T_s according to (4) and $N = \lfloor (l_{\mathcal{R}}/(D_s)) \rfloor$, implying a velocity- and range field length-dependent prediction horizon N. Any road information, such as road curvature, lane width, and so on, that may be required for the formulation of optimization problems will be evaluated (e.g., by linear interpolation) at the grid points $\{s_j\}_{j=0}^{N} = \hat{s}_k + \{0, D_s, \ldots, ND_s\}$, and subsequently denoted by subindex j. The generation of references z_j^{ref}, $\forall j = 1, \ldots, N$ and similarly for u_j^{ref} is discssued in Sections III-E and III-F. The LTV-MPC problem for ACC is defined as follows:

$$\min_{\substack{\{u_j\}_{j=0}^{N-1}, \\ \sigma, \sigma_{e_y}, \sigma_d}} \sum_{j=1}^{N-1} \|z_j - z_j^{\text{ref}}\|_{Q_z}^2 + \|z_N - z_N^{\text{ref}}\|_{Q_{zN}}^2 + \|\sigma_{e_y}\|_{q_{\sigma e_y}}^2 + \sum_{j=0}^{N-1} \|u_j - u_j^{\text{ref}}\|_{Q_u}^2 + \|u_j - u_{j-1}\|_{Q_{\Delta u}}^2 + \|\sigma_p\|_{Q_{\sigma_p}}^2 + \sum_{j=2}^{N-1} \|d_j - d_j^{\text{ref}}\|_{q_d}^2 + \|d_N - d_N^{\text{ref}}\|_{q_{dN}}^2 + \|\sigma_d\|_{q_{\sigma_d}}^2 \quad (7a)$$

$$\text{s.t.} \quad z_0 = \hat{z}_k \quad (7b)$$

$$u_{-1} = u_{k-1}^{\star} \quad (7c)$$

$$z_{j+1} = A_j z_j + B_j u_j + g_j, \quad j = 0, \ldots, N-1 \quad (7d)$$

$$p_{f,j} = l_{f,j} z_j + m_{f,j} u_j + n_{f,j}, \quad j = 1, \ldots, N \quad (7e)$$

$$p_{r,j} = l_{r,j} z_j + m_{r,j} u_j + n_{r,j}, \quad j = 1, \ldots, N \quad (7f)$$

$$u^{\min} \leq u_j \leq u^{\max}, \quad j = 0, \ldots, N-1 \quad (7g)$$

$$\Delta u^{\min} \leq u_j - u_{j-1} \leq \Delta u^{\max}, \quad j = 0, \ldots, N-1 \quad (7h)$$

$$e_{y,j}^{\min} - \sigma_{e_y} \leq e_{y,j} \leq e_{y,j}^{\max} + \sigma_{e_y}, \quad j = 1, \ldots, N \quad (7i)$$

$$p_{f,j}^{\min} - \sigma_{p,f} \leq p_{f,j} \leq p_{f,j}^{\max} + \sigma_{p,f}, \quad j = 1, \ldots, N \quad (7j)$$

$$p_{r,j}^{\min} - \sigma_{p,r} \leq p_{r,j} \leq p_{r,j}^{\max} + \sigma_{p,r}, \quad j = 1, \ldots, N \quad (7k)$$

$$\sigma_{e_y} \geq 0, \ \sigma = [\sigma_{p,f}, \sigma_{p,r}] \geq 0 \quad (7l)$$

$$d_0 = \hat{d}_k^{\text{obj},i^*}, \quad \text{where } i^* \in \{1, \ldots, N^{\text{obj}}\} \quad (7m)$$

$$d_{j+1} = d_j + a_{d,j} z_j + g_{d,j}, \quad j = 0, \ldots, N-1 \quad (7n)$$

$$d_j \geq d_{\min} + c_{\min} v_{x,j} - \sigma_d, \quad j = 1, \ldots, N \quad (7o)$$

$$\sigma_d \geq 0. \quad (7p)$$

The slack variables σ_{e_y}, $\sigma_{p,f}$, $\sigma_{p,r}$, and σ_d are used for constraint softening. Vector $u_{k-1}^{\star}$ denotes the input applied to the system at the previous sampling time. The parameters Q_z, Q_{zN}, Q_u, $Q_{\Delta u}$, Q_{σ_p}, $q_{\sigma_{ey}}$, q_{σ_d}, q_d, and q_{dN} are tuning weights. Safety parameters are $d_{\min}$ and $c_{\min}$. Constant upper and lower bounds are $u^{\min}$, $u^{\max}$, $\Delta u^{\min} = \dot{u}^{\min} T_s$, and $\Delta u^{\max} = \dot{u}^{\max} T_s$. In the OA, RT, and Brake modes, the weights q_{σ_d}, q_d, and q_{dN} are set to zero (i.e., the third row of the objective function is omitted) and the constraints (7m)–(7p) are removed. In case of ACC mode, $i^{\star}$ indicates the object which the ego car is meant to adapt its velocity to. This decision is relevant in case of multiple objects, including static ones. The closed-loop LTV-MPC system is not guaranteed to be stable. Recursive feasibility is guaranteed by construction as all state constraints are soft and the only hard constraints are those on inputs that we know we can always satisfy. To *encourage* stability, we added soft constraints (7e), (7f), (7j), and (7k) on front and rear tire slip angles to not enter the (strongly) nonlinear/unstable region of tire characteristics [23], [24]. For the provided vehicle model, lateral forces on front and rear tires, $F_{f,y}(u_\delta, v_y, \omega, v_x)$ and $F_{r,y}(v_x, v_y, \omega)$, are given as a linear function of the slip angles $\alpha_f(\omega, v_y, v_x)$ and $\alpha_r(\omega, v_y, v_x)$ with subsequent saturations at maximal lateral forces. Note that in contrast to the aforementioned two references, linearization and discretization need to be carried out in space coordinates when deriving (7e) and (7f) for our case. As a detail, our simple tire model sets $p_{h,j}^{\text{i}}$ constant for all $j = 1, \ldots, N$, $h \in \{f, r\}$ and $i \in \{\max, \min\}$. An alternative approach is adding a sufficient stability condition for LTV-MPC in form a quadratic constraint, which is, however, computationally significantly more expensive to solve [25].

E. Geometric Corridor Planning

An important task of the motion planning system is to ensure that the ego car travels inside the lane boundaries while safely avoiding all obstacles (*corridor planning*). The problem is nonconvex due to the fact that overtaking is possible on both sides of obstacles. It is further complicated because of a time-varying environment (with dynamically moving objects) requiring possibly frequent recursive replanning of corridors.

In [26], a convex polyhedral approximation approach for OA is compared with a hybrid MPC method [27]. For an approach based on mixed integer linear programming (see [28]). Motion planning using the vehicle's dynamical model and RRTs is described in [29]. Finding a shortest path on a visibility graph for motion planning is the technique used in [30]. A "hybrid" A* method for free form navigation is used in [31]. In [11], the high-level path planner employs dynamic programming (DP) to decide on which side to overtake the obstacles by solving a shortest path problem on a *spatial-temporal* grid; corridor constraints are then adjusted based on the optimal trajectory obtained from DP, before a library of motion primitives is used for path planning. In [7]–[10], at most two static obstacles are considered and heuristics are used to determine on which side to overtake.

The motivation for our geometric corridor planner is our spatial-based control approach and the desire to handle multiple obstacles, which we assume (for proof of concept) to be available via car-2-car communication. Indeed, as pointed out in [16], increases in lane capacity, e.g., through platooning of automated vehicles and the reduction in traffic congestions are expected to *not* be possible *without* car-2-car communication. An application, therefore, is our companion paper [13]. The need for a fast corridor planner, capable of treating many more than just one obstacle, becomes even more important in countries characterized by unorganized traffic, where vehicles of all types of sizes (from trucks to rickshaws) may travel anywhere inside road boundaries at arbitrary traveling speed, resulting in higher traffic bandwidth and significantly more overtaking [32].

At every sampling time along the corridor coordinate s, there needs to be a *velocity*- and *trajectory*-adjusted mapping of all objects. This adjustment is achieved by solving, for every object, the equation $s(t^\star) = s^{\text{obj}}(t^\star)$ for $t^\star$ and $s(t^\circ) = s^{\text{obj}}(t^\circ) + l^{\text{obj}}(t^\circ)$ for t°, where $s(t)$ is the ego car CoG position along the road centerline of the corridor at time t and similarly for the object. The velocity-adjusted object positions are then located between $s(t^\star)$ and $s(t^\circ)$. Including additional safety margins, they are used for graph generation (see Algorithm 3). In this paper, for the corridor planning, we model the progress of the ego car and other vehicles with constant velocities and thus obtain $s(t) = \hat{s}_k + (t - kT_s)\hat{v}_{x,k}$, for $t \geq kT_s$, and similarly for the objects. Then, we can derive

$$s(t^\star) = \hat{s}_k + \left(\frac{\hat{s}_k^{\text{obj}} - \hat{s}_k}{\hat{v}_{x,k} - \hat{v}_{x,k}^{\text{obj}}} \right) \hat{v}_{x,k} \tag{8}$$

$$s(t^\circ) = s(t^\star) + \frac{\hat{l}^{\text{obj}} \hat{v}_{x,k}}{\hat{v}_{x,k} - \hat{v}_{x,k}^{\text{obj}}}. \tag{9}$$

The lateral displacement with respect to road centerline can then be similarly evaluated as $e_y(t^\star)$ and $e_y(t^\circ)$.

We develop a path planner using graph theory for computational efficiency. To address the issue of trajectory feasibility, we follow the approach of heuristically augmenting objects by velocity-dependent safety margins chosen according to closed-loop driving simulation over the velocity range 30–130 km/h. We select lateral safety margins as a linear function of $\hat{v}_{x,k}$, such that at $\hat{v}_{x,k} = 130$ km/h, a safety distance of at least 2 m between the edge of the ego car and the boundary of another object is maintained. Selections for front and rear longitudinal safety distances are similarly linearly dependent on $\hat{v}_{x,k}$ and $\hat{v}_{x,k}^{\text{obj}}$. We further encourage an early start of steering on sight of an obstacle, thereby avoiding large incremental steering changes. Fundamental is the reachability of areas lateral of obstacles free for trespassing. Following the projection point of such areas ensures (geometrically) minimal heading variation. Possible trajectories can conveniently be modeled as a transition graph, with the absolute heading variations as edge weights. The trajectory that provides *cumulated minimal absolute heading variation* and safe avoidance of all obstacles can be considered for the adjustment of bounds on e_y. The complexity of the corridor planning problem increases with the number of obstacles. Assuming N^{obj} objects longitudinally displaced along a corridor, there are $2^{N^{\text{obj}}}$ possible combinations of overtaking. Our method naturally also allows for longitudinally overlapping (but laterally displaced) object constellations. It is outlined in Algorithm 2. For the creation of $e_y^{\min}$ and $e_y^{\max}$ in Step 8 of Algorithm 2, there exist different methods, such as stairwise or smoothed adaptation. Our preferred method is indicated in Fig. 4(b), making use of the s-coordinates of all objects. In Algorithm 2, the graph generation step is computationally and conceptually decisive. Modeling objects as rectangles aligned with the road centerline allows one to transition between nodes via two PWA lines (constant slope and constant e_y level). See Fig. 4(a) and (b) for illustration. For N_{nodes} different nodes, each node position $i \in \{1, \ldots, N_{\text{nodes}}\}$ is summarized in P_{list}

Algorithm 2 Corridor Planning

1: **Input**: corridor area, velocity- and safety margin-adjusted obstacle locations, initial ego car position and heading.
2: **Order** objects according to increasing s-position along the corridor and identify corresponding road bounds.
3: **Dismiss** objects significantly faster and in front or slower and behind the ego car position.
4: **Adjust** road bounds if objects overlap longitudinally with ego car CoG-position, or a neighboring lane is blocked.
5: **If** there still remain objects within the updated corridor:
6: Build a transition graph T and store the corresponding node positions in P_{list} using Algorithm 3.
7: Use the Label Correcting Algorithm (LCA) [33] to find the cumulatively least-heading-varying path $\bar{e}_y^{\text{ref}}$.
8: Given $\bar{e}_y^{\text{ref}}$ and the object locations, determine $e_y^{\min}$ and $e_y^{\max}$. See Figure 4(b) for illustration.
9: **Else**
10: Set $e_y^{\min}$ and $e_y^{\max}$ to the adjusted road bounds, set $\bar{e}_y^{\text{ref}}$ to the e_y-level of a designated road lane .
11: **End If**

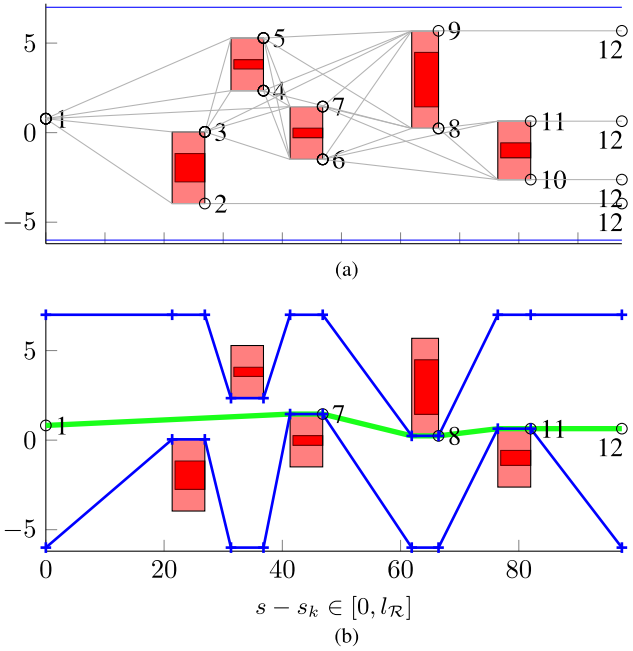

Fig. 4. Illustration of the geometric corridor planner. There are five trajectory-adjusted objects within a road-aligned corridor of length 100 m and original upper and lower road bounds $e_y^{\text{road,upper}} = 7$ and $e_y^{\text{road,lower}} = -6$. The red rectangles denote solid, in general, velocity-adjustedly mapped objects and the lighter red areas include the heuristically determined safety margins. The green PWA line illustrates the cumulatively least-heading-varying path determined on the graph. (a) Illustration of our graph-generation method, see Algorithm 3. (b) The solution returned by the geometric corridor planner.

by $P_{\text{list},i} = [s_i \; l_i \; e_{y,i}]$, where s_i denotes the s position where the constant e_y-level line part begins, l_i the length of this part, and $e_{y,i}$ the corresponding e_y level. For node 1 and the terminal node, we set $l = 0$. Then, a transition matrix $T \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{nodes}}}$ can be defined with entries

$$T_{ij} = \begin{cases} \Delta\theta_{ij}, & \text{if } \exists \text{ a PWA transition } i \to j \\ \infty, & \text{otherwise} \end{cases}$$

$$\Delta\theta_{ij} = \left| c - \tan^{-1}\left(\frac{e_{y,j} - e_{y,i}}{s_j - (s_i + l_i)} \right) \right| \tag{10}$$

where $c = \hat{e}_{\psi,k}$ for $i = 1$ and $c = 0$ for $i \in \{2, \ldots, N_{\text{nodes}}\}$.

TABLE I

ADAPTATION OF REFERENCE TRAJECTORIES v_x^{REF} AND e_y^{REF}, AND WEIGHT Q_z ACCORDING TO THE DRIVING MODE SELECTION, WHERE $j = 0, \ldots, N$. WE ABBREVIATE $\Delta = \hat{e}_{y,k} - e_y^{\text{ROAD}}$, WHERE e_y^{ROAD} DENOTES THE DESIGNATED REFERENCE LANE e_y LEVEL FOR OBSTACLE-FREE ROAD TRACKING (USUALLY $e_y^{\text{ROAD}} = 0$)

	1–'ACC'	2–'OA'	3–'RT'	4–'Brake'
$v_{x,j}^{\text{ref}}$	$\min(v_{x,j}^{\text{road}}, \hat{v}_{x,j}^{\text{obj},i^\star})$	$v_{x,j}^{\text{road}}$	$v_{x,j}^{\text{road}}$	$(0.2 + 0.8\frac{d_j^{\text{ref,obj},i^\star}}{l_{\mathcal{R}}})v_{x,j}^{\text{road}}$
$e_{y,j}^{\text{ref}}$	$\begin{cases} \bar{e}_{y,j}^{\text{ref}}, & \text{if } \bar{e}_y^{\text{ref}} \neq e_y^{\text{road}} \\ \tilde{e}_{y,j}^{\text{ref}}, & \text{if } \bar{e}_y^{\text{ref}} = e_y^{\text{road}} \ \&\ \lvert\Delta\rvert > 0.2 \\ e_y^{\text{road}}, & \text{otherwise} \end{cases}$	$\begin{cases} \bar{e}_{y,j}^{\text{ref}}, & \text{if } \bar{e}_y^{\text{ref}} \neq e_y^{\text{road}} \\ \tilde{e}_{y,j}^{\text{ref}}, & \text{if } \bar{e}_y^{\text{ref}} = e_y^{\text{road}} \ \&\ \lvert\Delta\rvert > 0.2 \\ 0, & \text{otherwise} \end{cases}$	$\begin{cases} \tilde{e}_{y,j}^{\text{ref}}, & \text{if } \lvert\Delta\rvert > 0.2 \\ e_y^{\text{road}}, & \text{otherwise} \end{cases}$	$\begin{cases} \tilde{e}_{y,j}^{\text{ref}}, & \text{if } \lvert\Delta\rvert > 0.2 \\ e_y^{\text{road}}, & \text{otherwise} \end{cases}$
Q_z	$\begin{bmatrix} Q_z(1,1) \\ Q_z(5,5) \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$	$\begin{bmatrix} Q_z(1,1) \\ Q_z(5,5) \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \end{bmatrix}$	$\begin{bmatrix} Q_z(1,1) \\ Q_z(5,5) \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$	$\begin{bmatrix} Q_z(1,1) \\ Q_z(5,5) \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$

Algorithm 3 Graph Creation for Corridor Planning

Preparation: given *velocity*- and *trajectory*-adjusted object positions within the corridor, fill $P_{\text{list}} \in \mathbb{R}^{(1+2N^{\text{obj}})\times 3}$ using the starting node and both boundary positions of each object.
Initialize: tree $T \in \mathbb{R}^{(2+2N^{\text{obj}})\times(2+2N^{\text{obj}})}$ with $T_{j_t j_t} \leftarrow 0$ where $j_t = (2 + 2N^{\text{obj}})$ is the terminal node ($s = l_{\mathcal{R}}$ but potentially variable e_y-positions), and list OPEN $\leftarrow \{1\}$.
While list OPEN is not empty:
  Remove a node i from OPEN and determine the corresponding s_i and $e_{y,i}$ from P_{list}.
  If there exists a direct path from s_i to $s = l_{\mathcal{R}}$ on $e_y = e_{y,i}$ without intersecting any other object:
    Set $T_{ij_t} \leftarrow 0$.
  Else
    Identify all objects ahead with $s > s_i$.
    For each object in front:
      Find corresponding two object boundary nodes j_1 ('left' of object) and j_2 ('right' of object).
      If transition from node i to j_1 is not intersecting with any other object:
        Compute $\Delta\theta_{ij_1}$, see (10).
        If $\Delta\theta_{ij_1} < \Delta\theta^{\max}$: $T_{ij_1} \leftarrow \Delta\theta_{ij_1}$ **End If**
        If $s_{j_1} + l_{j_1} \geq l_{\mathcal{R}}$:
          $T_{j_1 j_t} \leftarrow 0$.
        Else
          Add node j_1 to OPEN if it is not already in OPEN.
        End If
      End If
      Do the same operations for the transition from node i to j_2 as for from i to j_1.
    End For
  End If
End While
Remove any nodes that were never reached from T and P_{list}.

A characteristic behavior of the devised path planner is to start steering early, i.e., "on sight" of an obstacle. This is beneficial, since it results in minimal incremental steering changes, while still allowing for a quick heading correction if required. Importantly, a timely OA-maneuver initiation provides an early (partial) insight into a neighboring lane, and thereby may allow for detection of previously shielded objects if there is no car-2-car communication, or, in case of car-2-car communication, visual confirmation of the communicating vehicles. The PWA reference trajectories returned by the corridor planner are either directly fed to the spatial-based predictive controller, or first additionally smoothed according to our method [12]. In the following, the two reference trajectory schemes are denoted by "PWAref" and "Sref."

F. Adaptation of Reference Trajectories

In Table I, we define the body-frame reference velocities to be used in problem (7). Corresponding to $v_{x,j}^{\text{ref}}$, we adjust $\omega_{f,j}^{\text{ref}} = v_{x,j}^{\text{ref}}/r_t$ and $\omega_{r,j}^{\text{ref}} = v_{x,j}^{\text{ref}}/r_t$. As motivated in [7], we define $\omega_j^{\text{ref}} = \psi_{s,j}^{\text{ref}'} \cdot v_{x,j}^{\text{ref}}$. Furthermore, $v_{y,j}^{\text{ref}} = 0$ and $e_{\psi,j}^{\text{ref}} = 0$. For the input references, we set $u_j^{\text{ref}} = 0$, which yielded overall better performance than using optimal input trajectories from the last sampling time as references for the current ones. Vectors $e_{y,j}^{\min}$ and $e_{y,j}^{\max}$ are obtained from the corridor planning algorithm. For PWAref, we define

$$\tilde{e}_{y,j}^{\text{ref}} = \begin{cases} \hat{e}_{y,k} + \left(\dfrac{e_y^{\text{road}} - \hat{e}_{y,k}}{2}\right)\frac{j}{3}, & j = 0, \ldots, 3 \\ \dfrac{\hat{e}_{y,k} + e_y^{\text{road}}}{2} + \left(\dfrac{e_y^{\text{road}} - \hat{e}_{y,k}}{2}\right)\dfrac{j-3}{N-3}, & j = 4, \ldots, N \end{cases} \tag{11}$$

as a heuristic to mitigate overshooting when returning to the road centerline. An alternative is presented in [12] for case "Sref." Distance dynamics with respect to a leading object are relevant in the ACC mode. The velocity-dependent reference distance is defined as

$$d_j^{\text{ref}} = p_{\text{dref}}\hat{v}_{x,k} + d_{\min}, \quad j = 2, \ldots, N \tag{12}$$

where $d_{\min}$ and p_{dref} are calibrated. For example, $d_{\min} = 2$m and $p_{\text{dref}} = (27 - d_{\min})/(50/3.6)$ enforce a safety distance of 27 m at 50-km/h vehicle speed. This corresponds to the rule of thumb to keep a safety distance in meters of "slightly more than half of the speedometer indication in km/h." In order to not necessarily accelerate the ego car to achieve this distance, we set $d_j^{\text{ref}} = 0.95\hat{d}_j^{\text{obj},i^\star} + 0.05 d_j^{\text{ref}}$ if $d_j^{\text{ref}} < \hat{d}_j^{\text{obj},i^\star}$. This safety-oriented strategy is appropriate for autonomous driving. In contrast, for cooperative driving (with enhanced predictability of neighboring communicating cars), we may use (12) directly as is in order to very quickly achieve platooning [13].

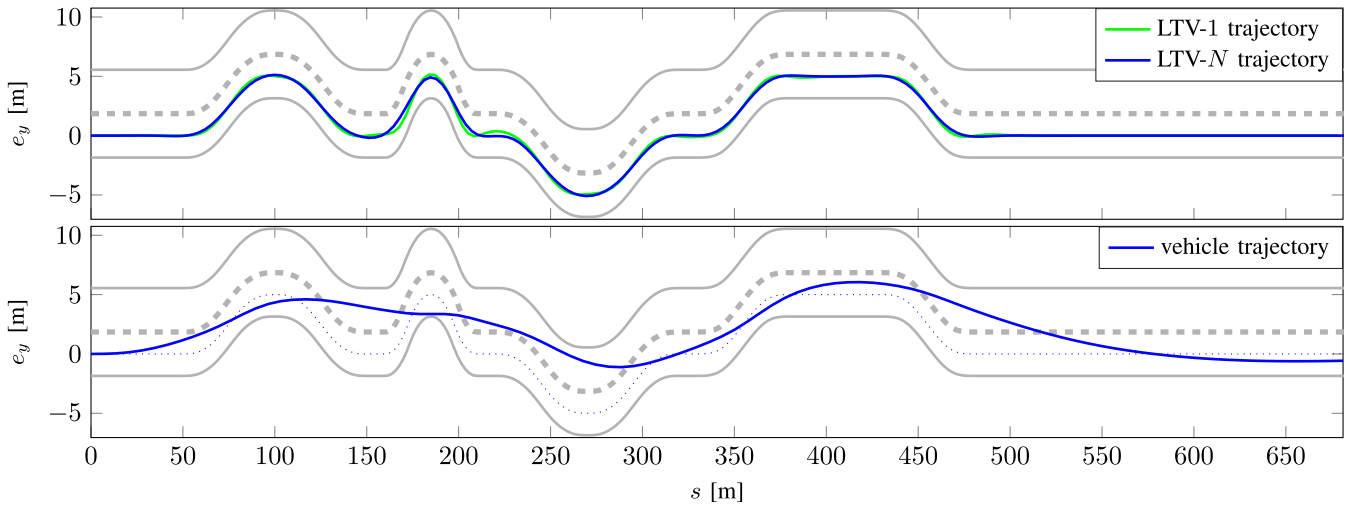

Fig. 5. Results of two obstacle-free road-profile tracking experiments. Top: reference velocity is 50 km/h. Bottom: change of tuning parameters in (7a) for high velocity traversal at approximately 100 km/h.

IV. Numerical Simulations

In this paper, four simulations are considered. Additional simulations are provided in [13]. We initialize the MPC weights as $Q_z = \text{diag}[100, 0, 0, 1, 10, 0, 0]$, $Q_{zN} = 10Q_z$, $Q_u = \text{diag}[0.01, 10, 100]$, $Q_{\Delta u} = \text{diag}[10, 100, 1000]$, $Q_{\sigma_p} = \text{diag}[10\,000, 10\,000]$, $q_{\sigma_{e_y}} = q_{\sigma_d} = 10\,000$, and $q_d = q_{dN} = 100$, and, subsequently, update them according to Table I. All simulations are conducted on a laptop running Ubuntu 14.04 equipped with an Intel Core i7 CPU at 2.80 GHz × 8, 15.6 GB of memory, and using MATLAB 9.1 (R2016b). Computation times are summarized in Table II. System states are integrated forward from the original nonlinear vehicle model including its lookup tables and using MATLAB's `ode23tb`. Selected animated simulations are available at http://dysco.imtlucca.it/mogens/sim_autonomous_driving.htm.

A. Road-Profile Tracking

For verification of road-profile tracking capabilities (RT-mode), a test track composed of straights and multiple bielementary paths of variable sharpness and symmetric point fractions was created. In a first experiment, we employed the same controller, which we also used in Sections IV-B–IV-D. For a comparison, instead of linearizing along the reference trajectory (solution denoted by LTV-N), we also maintained state-transition matrices and vectors constant over the prediction horizon ($A_j = A_0$, $\forall j = 0, \ldots, N-1$ and similarly for B_j, g_j, and so on) and denote this solution by LTV-1. Time $\bar{\tau}_{\text{qp,build}}$ was thereby reduced from 10.7 to 5.5 ms (the remaining times were unchanged). In a second experiment, we changed our control objective to traversing the road segment at 100 km/h without any emphasis on tracking the centerline of the road. We, therefore, changed the weights in the objective function (7a). In particular, we set $Q_z(5, 5) = 0$ to allow arbitrary e_y positions of the vehicle within the road bounds. In addition, penalties on steering were increased. Results are displayed in Fig. 5.

TABLE II

Average Computation Times $\bar{\tau}$ in Milliseconds. Time $\bar{\tau}_{\text{QP,BUILD}}$ Includes Linearization, Discretization, and Building of the QP (via the Elimination of States). Computation Times Using MATLAB's `quadprog` for the Solution of the QPs Are Denoted by $\bar{\tau}_{\text{QUADPROG}}$. For the Graph Creation, See Algorithm 3, and for the Subsequent Finding of a Cumulatively Least-Heading Varying Path on the Tree via LCA [33], $\bar{\tau}_{\text{TREE,BUILD}}$ and $\bar{\tau}_{\text{LCA}}$ Resulted. Time τ_{SMOOTH} Comprises Smoothing [12]. The State Estimator Time is τ_{DEKF}. The Average (Velocity-Dependent) Prediction Horizon is $\bar{N}$. As Discussed in Each Section, Results for Two Settings of v^{REF}, $l_{\mathcal{R}}$, or e_y^{REF} Are Reported. Corresponding Parameter Values (param.) Are Stated

	Sect. IV-A v^{ref} [km/h]	Sect. IV-B $l_{\mathcal{R}}$ [m]	Sect. IV-C e_y^{ref}-type	Sect. IV-D e_y^{ref}-type
param.	50/100	50/100	PWA/Sref	PWA/Sref
$\bar{\tau}_{\text{qp,build}}$	10.7/5.6	5.4/11.0	5.1/5.1	17.0/16.7
$\bar{\tau}_{\text{quadprog}}$	56.3/17.0	16.1/41.6	14.5/14.5	85.0/85.0
$\bar{\tau}_{\text{tree,build}}$	-	0.8/0.8	0.7/0.7	1.0/1.1
$\bar{\tau}_{\text{LCA}}$	-	0.4/0.3	0.2/0.2	0.2/0.2
$\bar{\tau}_{\text{smooth}}$	-	15.2/15.1	-/6.1	-/6.9
$\bar{\tau}_{\text{dEKF}}$	1.1/1.2	1.6/1.8	1.1/1.2	1.1/1.1
$\bar{N}$	29/14	14/28	12/12	38/38

B. Avoidance of Two Static Obstacles

Closed-loop simulation results for the overtaking of two static obstacles are displayed in Fig. 6. Setting $e_y^{\text{ref}} = 0$ for road centerline tracking indicates the case when using the corridor planner only for constraint selection. PWAref and Sref-solution denote resulting closed-loop vehicle paths when explicitly employing nonzero reference trajectories in the objective function (7a) of the LTV-MPC problem. The advantage of an explicit reference trajectory is apparent from the example with $l_{\mathcal{R}} = 100$: even though the second obstacle is visible, a controller with $e_y^{\text{ref}} = 0$ would conduct an unnecessary steering maneuver.

At *every* T_s, the geometric path planner solves the corridor planning problem and outputs a PWA reference trajectory e_y^{ref}. This can either be fed directly to the spatial-based controller or first processed in an additional smoothing step. As outlined earlier, the corresponding reference trajectories are referred to

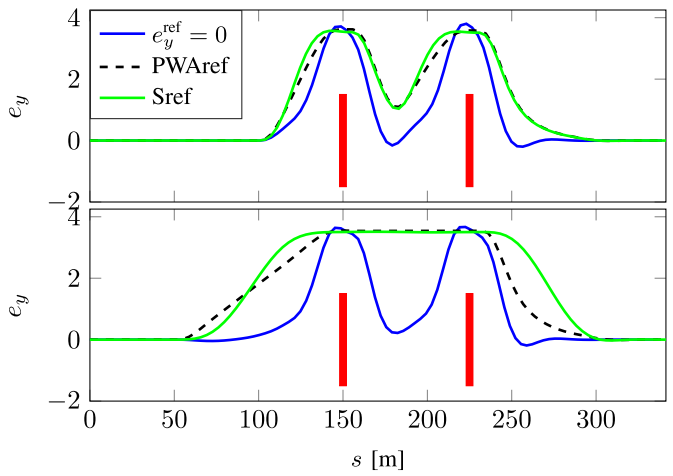

Fig. 6. Two static obstacles simulation. The red rectangles denote static objects. The entry speed of the ego car is 50 km/h. The range view at front, $l_{\mathcal{R}}$, is 50 m (top) and 100 m (bottom). We test different methods for the reference trajectory generation. The beneficial effect of using an explicit path planner making use of all available information is apparent from the bottom frame.

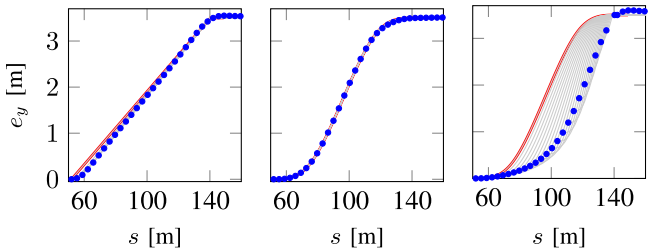

Fig. 7. Influences of recursive replanning of reference trajectories. The red thickened line indicates the trajectory planned upon detection of the obstacle. The blue dots indicate the actual closed-loop trajectory traveled by the vehicle. The additional thin lines indicate any replanned reference trajectories *after* first object detection. Left frame: PWAref, with replanning at *every* T_s. Middle frame: Sref according to [12], explicitly *not* replanning at every T_s. Right frame: clothoid-based replanning at *every* T_s. See also Fig. 8.

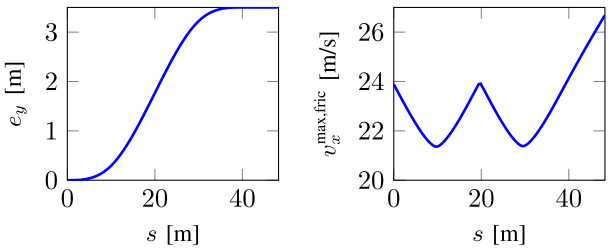

Fig. 8. Illustration of a maximal admissible friction speed profile. Speed profile minima are obtained at maximal curvature along the path trajectory.

as PWAref and Sref, respectively. The influence of recursively replanning obstacle avoiding reference paths is visualized in Fig. 7 by means of a static obstacle.[1] Despite the recursive replanning at *every* T_s for PWAref, it does only minimally effect the closed-loop trajectory following the original reference trajectory planned upon first object sight. This is in stark contrast to when replanning at *every* T_s based on clothoids.

[1]Note that for a static OA maneuver, any recursively *replanned* trajectories should ideally coincide with the original obstacle avoiding trajectory planned upon first obstacle sight.

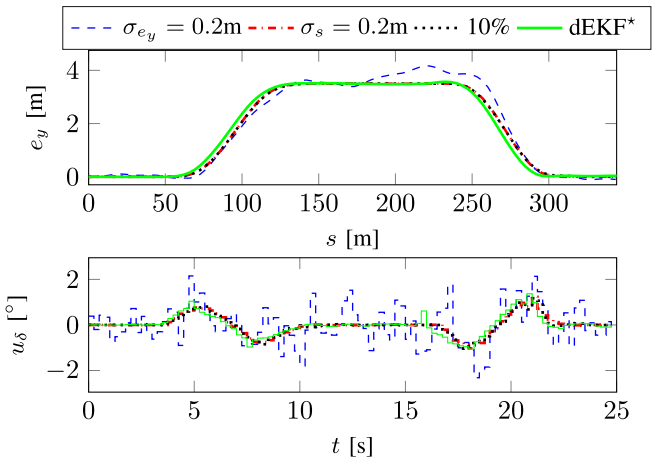

Fig. 9. For four OA experiments (two static obstacles, $l_{\mathcal{R}} = 100$m), the closed-loop vehicle trajectories and corresponding steering input are shown. In the first two experiments, lateral and longitudinal measurements are perturbed by measurement noise with zero mean and a standard deviation of 0.2 m, respectively. In the third experiments, at every T_s, 10% of any of the measurements of state vector $z_t \in \mathbb{R}^8$ are simulated to be randomly missing. For the first three experiments, the dEKF is tuned, such that if a measurement is available, this measurement is returned as state estimate. In the final experiment, perturbations of all of the previous experiments act simultaneously, and the dEKF is now allowed model-based estimation.

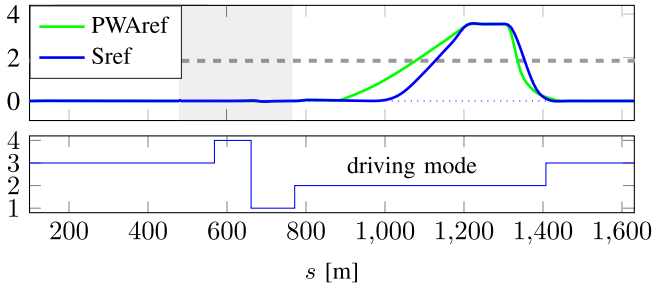

Fig. 10. Highway simulation. The gray area, when reached by the ego car, indicates the left lane being blocked, and thus prohibiting a lane change, due to the fast vehicle advancing from behind. The OA maneuver, i.e., the overtaking of the slower vehicle, is initiated once the left lane is cleared.

The experiments of Fig. 9 were used to validate the dEKF design. The first experiment is further meant to emphasize the importance of accurate lateral positioning sensors.

C. Highway Simulation

A typical driving scenario on a highway is described. There is a faster car advancing from behind on the left lane and there is a slower car ahead on the same lane of the ego car. See Figs. 10 and 11 for simulation results. All four driving modes are activated for resolving the given object constellation.

D. Multiple Obstacles Simulation

The simulation comprises a two-lane road, multiple static obstacles, one vehicle advancing from the front on the left lane, a sinuous road section, and one leading vehicle, first detected after the 480-m mark. The general road reference velocity is 50 km/h. When the ego car approximately reaches the 280-m mark, it occurs that the left road lane becomes blocked due to an approaching vehicle. As a consequence,

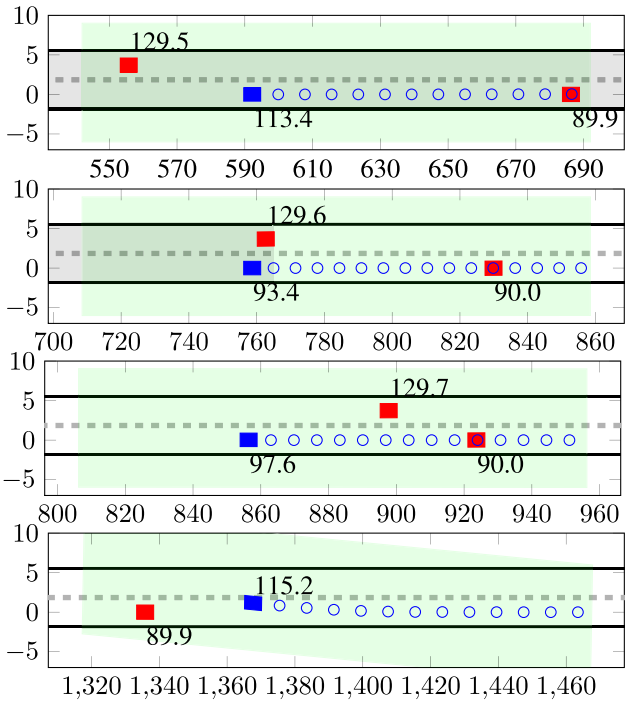

Fig. 11. Highway simulation. The velocities [km/h] of the objects (red) and the ego car (blue) are indicated next to them. The blue dots indicate the reference trajectory of the ego car along the prediction horizon. The green area shows the range field in which objects can be detected. The x- and y-axes indicate position coordinates in meters. (a) Due to a fast vehicle advancing from behind, the controller prohibits a lange change ("blocks" the left lane). Simultaneously, an object with slower velocity is detected in front and the braking mode is triggered. (b) Ego car has decelerated its velocity sufficiently, and is in ACC mode, adjusting its velocity to the leading vehicle. (c) Faster car has passed the ego car, the left lane is free for overtaking, and an OA maneuver (OA-mode) is initiated in order for the ego car to pursue its set reference velocity of 115 km/h. (d) Ego car has overtaken the slower vehicle and is returning to road-profile tracking (RT-mode).

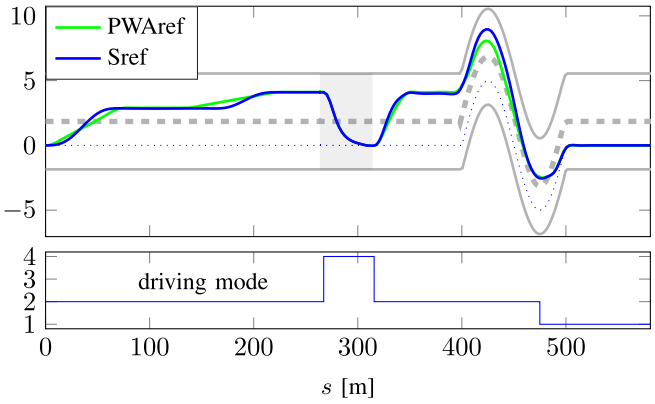

Fig. 12. Multiple obstacles simulation. The closed-loop trajectories when using PWAref and Sref references. The gray area, when reached by the ego car, indicates the left lane being blocked due to an advancing vehicle from the front.

the braking mode is activated and a lane change to the right lane is performed. The corresponding vehicle trajectory is given in Fig. 12. Six snapshots of the complete simulation are shown in Fig. 13. The sinuous road segment is handled naturally by the formulation of the LTV-MPC in a road-aligned coordinate frame.

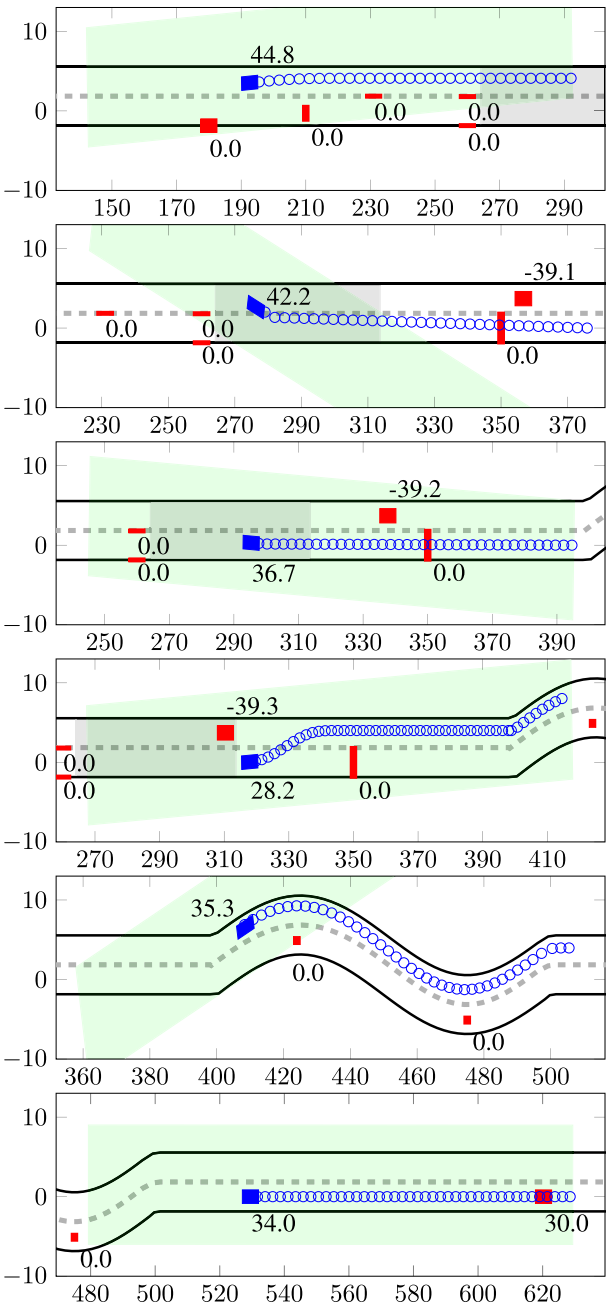

Fig. 13. Multiple obstacles simulation. The velocities [km/h] of the objects (red) and the ego car (blue) are indicated next to them. The blue dots indicate the reference trajectory of the ego car along the prediction horizon. The green area indicates the range field in which objects are detectable. The x- and y-axes indicate position coordinates in meters. The light gray region denotes an area, which, when reached by the ego car, implies that the left lane is blocked. Here, due to an advancing car on the left lane. (a) Ego car is in OA-mode overtaking multiple static objects. (b) Ego car has detected a vehicle advancing on the left lane with an estimated velocity of 39.1 km/h, and, because of the additional static obstacle on the right lane around the 350m-mark, subsequently activated the Brake mode. (c) Detected vehicle is still coming *toward* the ego car with velocity 39.2 km/h. The left lane is thus still blocked and the braking mode activated. (d) Vehicle on the left lane has passed. The left lane is free and a static obstacle on the right lane must be avoided. (e) There are two static obstacles on a curved road segment. (f) Ego car detects a vehicle in front and already starts to adjust its velocity accordingly (ACC mode).

V. Conclusion

We presented a general control framework for autonomous driving whose main component is a deterministic LTV-MPC

in a road-aligned coordinate frame for reference trajectory tracking of PWA lines generated by a geometric path planner using graphs to solve the combinatorial corridor planning problem. PWA reference trajectories can additionally be smoothed following our other work [12]. The coupling of ACC and OA enabled handling of driving scenarios with dynamic objects. Sensor uncertainty and missing measurements were dealt with by a dEKF. We discussed switching rules for four driving modes, the importance of velocity-adjusted obstacle positions within the corridors, the role of safety margins, and discussed the role of recursive replanning of reference trajectories. The presented control systems framework is equally suitable for autonomous as well as cooperative vehicle automation systems, where every one of the automated vehicles may be equipped with the control architecture presented here [13]. Fundamental for cooperative driving is the coupling of ACC with OA capabilities. Ultimately, we motivated that distinguishing between two vehicle models, one dynamic and one kinematic for high and low (close to zero) velocities, permits to conduct control design entirely spatial based for the whole vehicle operating range (from parking to high speed OA). For future work, more elaborate tire dynamics need to be incorporated into the vehicle model to justify the usage of smoothed reference trajectories. Second, safety margin heuristics and switching rules can be identified by machine learning from manual driving. Ultimately, experimental validation on a vehicle is sought.

REFERENCES

[1] V. L. Bageshwar, W. L. Garrard, and R. Rajamani, "Model predictive control of transitional maneuvers for adaptive cruise control vehicles," *IEEE Trans. Veh. Technol.*, vol. 53, no. 5, pp. 1573–1585, Sep. 2004.

[2] R. Möbus, M. Baotic, and M. Morari, *Multi-Object Adaptive Cruise Control*. Berlin, Germany: Springer, 2003.

[3] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 556–566, May 2011.

[4] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 707–714, May 2011.

[5] S. Lefèvre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *Proc. IEEE Intell. Vehicles Symp.*, Jun./Jul. 2015, pp. 920–926.

[6] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Proc. ASME Dyn. Syst. Control Conf.*, 2010, pp. 265–272.

[7] Y. Gao *et al.*, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proc. 11th Int. Symp. Adv. Vehicle Control*, 2012, pp. 1–6.

[8] A. Gray, M. Ali, Y. Gao, J. K. Hedrick, and F. Borrelli, "Semi-autonomous vehicle control for road departure and obstacle avoidance," in *Proc. IFAC Control Transp. Syst.*, 2012, pp. 1–6.

[9] J. V. Frasch *et al.*, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. IEEE Eur. Control Conf.*, Jul. 2013, pp. 4136–4141.

[10] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *Proc. IEEE Amer. Control Conf.*, Jun. 2012, pp. 4239–4244.

[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Sep./Oct. 2014.

[12] M. G. Plessen and A. Bemporad, "Recursive trajectory planning for autonomous vehicles using generalized elementary paths," in *Proc. IFAC World Congr.*, 2017.

[13] M. G. Plessen, D. Bernardini, H. Esen, and A. Bemporad, "Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one- and bi-directional traffic flow control," in *Proc. IEEE Conf. Decision Control*, Dec. 2016, pp. 1582–1588.

[14] C. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Trans. Autom. Control*, vol. 23, no. 3, pp. 395–404, Jun. 1978.

[15] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2011.

[16] S. E. Shladover, "Cooperative (rather than autonomous) vehicle-highway automation systems," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 1, pp. 10–19, Jan. 2009.

[17] E. Guizzo, "How Google's self-driving car works," *IEEE Spectr.*, Oct. 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[18] N. Shchetko, "Laser eyes pose price hurdle for driverless cars," *Wall Street J.*, Jul. 2014. [Online]. Available: https://www.wsj.com/articles/laser-eyes-pose-price-hurdle-for-driverless-cars-1405969441

[19] M. D. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode, "Location-aware services over vehicular ad-hoc networks using car-to-car communication," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 8, pp. 1590–1602, Oct. 2007.

[20] Y. Song and J. W. Grizzle, "The extended Kalman filter as a local asymptotic observer for nonlinear discrete-time systems," in *Proc. IEEE Amer. Control Conf.*, Jun. 1992, pp. 3365–3369.

[21] B. D. Anderson and J. B. Moore, *Optimal Filtering*. Chelmsford, MA, USA: Courier Corporation, 2012.

[22] P. C. Young, *Recursive Estimation and Time-Series Analysis: An Introduction for the Student and Practitioner*. Berlin, Germany: Springer, 2012.

[23] 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, May 2007.

[24] A. Katriniok and D. Abel, "LTV-MPC approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics," in *Proc. 50th IEEE Conf. Decision Control Eur. Control Conf.*, Dec. 2011, pp. 6828–6833.

[25] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 862–875, May 2008.

[26] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *Proc. IEEE Decision Control Eur. Control Conf.*, Dec. 2011, pp. 7488–7493.

[27] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," in *Proc. 18th IFAC World Congr.*, Jan. 2011, pp. 11900–11906.

[28] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. IEEE Eur. Control Conf.*, Sep. 2001, pp. 2603–2608.

[29] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 1681–1686.

[30] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.

[31] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

[32] R. Kala and K. Warwick, "Planning autonomous vehicles in the absence of speed lanes using an elastic strip," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1743–1752, Dec. 2013.

[33] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA: Athena Scientific, 1995.

[34] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 2061–2067.

[35] A. Carvalho, S. Lefèvre, 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, Jul. 2015.

[36] P. Falcone, M. Ali, and J. Sjöberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1352–1361, Dec. 2011.

[37] J.-C. Latombe, *Robot Motion Planning*, vol. 124. New York, NY, USA: Springer, 2012.

[38] J. Funke and J. C. Gerdes, "Simple clothoid lane change trajectories for automated vehicles incorporating friction constraints," *J. Dyn. Syst., Meas., Control*, vol. 138, no. 2, p. 021002, Dec. 2016.
[39] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.
[40] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager, "Optimal control of formula 1 race cars in a vdrift based virtual environment," in *Proc. 18th IFAC World Congr.*, vol. 18. 2011, pp. 11907–11912.
[41] P. A. Ioannou and C. C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. Veh. Technol.*, vol. 42, no. 4, pp. 657–672, Nov. 1993.
[42] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Oct. 2013, pp. 2335–2340.
[43] J. Funke *et al.*, "Up to the limits: Autonomous Audi TTS," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2012, pp. 541–547.

Mogens Graf Plessen received the master's degree in mechanical engineering from ETH Zürich, Zürich, Switzerland, in 2014 . He is currently pursuing the Ph.D. degree in control systems with IMT Institute for Advanced Studies Lucca, Lucca, Italy.

His current research interests include autonomous driving, finance, and logistics.

Daniele Bernardini received the master's degree in computer engineering and the Ph.D. degree in information engineering from the University of Siena, Siena, Italy, with a major in automatic control, in 2007 and 2011, respectively.

From 2011 to 2015, he was a Post-Doctoral Fellow with IMT Lucca, Lucca, Italy. In 2011, he co-founded ODYS Srl, Lucca, a company specialized in the development of model predictive control software for embedded systems. His current research interests include MPC, stochastic control, networked control systems, hybrid systems, and their application to real-time problems in the automotive, aerospace, and energy domains.

Hasan Esen received the master's degree in mechatronics from the Technical University of Hamburg-Harburg, Hamburg, Germany, in 2003, and the Ph.D. degree in control engineering from the Technical University of Munich, Munich, Germany, in 2007.

He held a research assistant position at the Technical University of Munich. Since 2009, he has been a researcher with the Corporate Research and Development Department, DENSO AUTOMOTIVE Deutschland GmbH, Eching, Germany. He is currently a Technical Manager and the Leader of the Advanced Technology Research and Development Team activities, which mainly cover advanced control, networked control, cyber-physical system analysis, and model-based system engineering domains. He conducts academic university collaboration projects, and participates in public funded projects.

Alberto Bemporad (F'10) received the master's degree in electrical engineering and the Ph.D. degree in control engineering from the University of Florence, Florence, Italy, in 1993 and 1997, respectively.

From 1996 to 1997, he was with the Center for Robotics and Automation, Department of Systems Science and Mathematics, Washington University, St. Louis, MO, USA. From 1997 to 1999, he held a post-doctoral position at the Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland, where he collaborated as a Senior Researcher until 2002. From 1999 to 2009, he was with the Department of Information Engineering, University of Siena, Siena, Italy, becoming an Associate Professor in 2005. From 2010 to 2011, he was with the Department of Mechanical and Structural Engineering, University of Trento, Trento, Italy. Since 2011, he has been a Full Professor with the IMT Institute for Advanced Studies, Lucca, Italy, where he served as the Director of the institute from 2012 to 2015. He spent visiting periods at the University of Michigan, Ann Arbor, MI, USA, and Zhejiang University, Hangzhou, China. In 2011, he Co-Founded ODYS S.r.l., a consulting and software development company specialized in advanced controls and embedded optimization algorithms. He has authored over 300 papers in the areas of model predictive control, automotive control, hybrid systems, multiparametric optimization, computational geometry, robotics, and finance, and co-inventor of seven patents. He is an Author or Co-Author of various MATLAB toolboxes for model predictive control design, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox.

Dr. Bemporad received the IFAC High-Impact Paper Award for the 2011-14 triennial. He was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL from 2001 to 2004 and the Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society from 2002 to 2010.