Predictive Guidance and Control Framework for (Semi-)Autonomous Vehicles in Public Traffic

Thomas Weiskircher, Qian Wang, and Beshah Ayalew, *Member, IEEE*

***Abstract*— In this paper, a predictive trajectory guidance and control framework is proposed that enables the safe operation of autonomous and semiautonomous vehicles considering the constraints of operating in dynamic public traffic. The core module of the framework is a nonlinear model predictive guidance module that uses a computationally expedient curvilinear frame for the description of the road and of the motion of the vehicle and other objects. The module enforces constraints generated from information about obstacles/other vehicles/objects, public traffic rules for speed limits and lane boundaries, and the limits of the vehicle's dynamics. The module can be configured in two basic modes. The first is a tracking mode, where the control inputs computed by the model predictive guidance module act as references for traditional lower level control systems. The second is a planning mode, where the traffic-optimal state trajectories computed by the model predictive control are reinterpreted for planning the optimal path and speed, which in turn can be tracked by an elaborate speed and path tracking controller. The performance of most aspects of the proposed scheme is illustrated by considering various simulations of the control framework applied to a high-fidelity vehicle dynamics model of the (semi-)autonomous vehicle in typical public driving events, such as intersections, passing, emergency braking, and collision avoidance. The feasibility of the proposed control framework for real-time application is highlighted with the discussions of the computational execution times observed for these various scenarios.**

***Index Terms*— Autonomous vehicles, model predictive control (MPC), public traffic constraints, semiautonomous vehicles.**

I. Introduction

AUTONOMOUS and semiautonomous vehicles have a huge potential for boosting the efficiency of road transportation systems via safe increases in traffic density, minimizing pollutions, and energy waste. As such, they are currently under active research and development. A primary research area for these vehicles is the design of robust and computationally feasible control frameworks that guarantee a collision-free trajectory guidance of the vehicles under the constraints of prevailing road boundaries, other objects, and traffic regulations.

The majority of the existing studies in trajectory planning and guidance are found in the robotics field, where various algorithms are proposed to find collision-free trajectories under static and dynamic constraints in the available space. In this discussion, the word trajectory is used to mean state trajectories (including, at a minimum, both position or path and speed) for the controlled robot/vehicle. The state of the art in planning methods roughly falls into three groups. The first are the sampling-based methods where the state and/or input space is discretized or randomly sampled in lattices and then efficient heuristics for deterministic or stochastic searching, such as the A* graph search or Rapidly Exploring Random Tree* algorithm is applied to find the collision-free trajectory based on an objective function [1]–[3]. However, the existence and optimality of the solution depends on the size of the lattice; improves with lattice size but comes at the cost of higher computational burden. A second group implements decoupling schemes for planning the global path and for the calculation of the speed for local obstacle avoidance [4]. This is also a useful separation for passenger vehicles on public roads, since the planning of the global path (often used to mean navigation or route finding via a GPS-based navigation system employing street maps) and replanning in case that a collision is eminent locally. For the problem of replanning state trajectories near dynamic obstacles, special algorithms have been proposed for specific maneuvers, such as lane change and obstacle avoidance [5], [6]. The disadvantage of such maneuver specific planning algorithms is the added need for a maneuver detection and coordination scheme. A third group of planning algorithms involve mathematical constrained optimization formulations which offer some guarantees of conditional existence and optimality of the solution based on the convexity of the problem formulation and the quality of initial guesses [7], [8].

Model predictive control (MPC), which belongs to the third group, offers a convenient and optimal replanning method that can accommodate dynamic changes in the environment of the controlled vehicle via frequent updates in its receding horizon implementation. Although MPC could also involve large computational burdens as it typically solves a linear or nonlinear optimization problem at each control update, recent developments in the field of real-time optimization algorithms

Manuscript received March 31, 2016; revised August 19, 2016; accepted December 2, 2016. Manuscript received in final form December 18, 2016. The work of T. Weiskircher was supported by a Post-Doctoral Fellowship from the German Academic Exchange Service (DAAD). Recommended by Associate Editor C. Canudas-de-Wit.

T. Weiskircher is with Daimler Research and Development AG, 70546 Stuttgart, Germany and also with the Applied Dynamics and Control Research Group, International Center for Automotive Research, Clemson University, Greenville, SC 29607 USA (e-mail: tweiski@clemson.edu).

Q. Wang and B. Ayalew are with the Applied Dynamics and Control Research Group, International Center for Automotive Research, Clemson University, Greenville, SC 29607 USA (e-mail: qwang8@clemson.edu; beshah@clemson.edu).

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.2016.2642164

have made it a feasible alternative [9], [10]. In this paper, we exploit these developments to propose a control framework for predictively guiding a road vehicle in the presence of multiple dynamic public traffic constraints.

In the last decade, there has been an extensive volume of work in the application of MPC to vehicle dynamics control. Some works focus on collision avoidance (CA) for (semi-) autonomous systems via single control inputs, such as active front steering [11], [12]. MPC has also been applied for lower level trajectory tracking [13], [14]. Some works combine a simplified trajectory planning module with a trajectory tracking module that uses high-fidelity nonlinear vehicle dynamics models in the MPC. These models require a large set of vehicle parameters (e.g., tire data, vehicle mass, and inertia properties, which are subject to change in operation). They also require high update rates consistent with the dynamics. To reduce the computational requirements, linearized models are often used within the MPC [15], [16]. While the execution time may be reduced with such linearization, these assumptions do not necessarily hold during dynamic events and for vehicles requiring combined lateral and longitudinal control over long prediction horizons; the results are at the best suboptimal.

The key consideration for designing a real-time implementable control framework for (semi-)autonomous vehicles is in the different time scales involved in the vehicle system dynamics. While high-bandwidth actuators (e.g., steering, motors and brakes) require a control sampling of 0.001–0.01 s, vehicle dynamics control is typically executed with a sampling time of 0.01–0.02 s. Trajectory planning *and* control (or *trajectory guidance*, for short) is a high-level function that needs to interface with these lower level vehicle dynamics controllers. The authors estimate that feasible sampling times for MPC-based trajectory guidance are in the range of 0.05–0.15 s, with longer times needed for (semi-)autonomous driving in public traffic featuring dynamic obstacles and complex traffic constraints. The computational burden of the nonlinear programming problem involved increases with the model order or number of states, the number of control inputs, the nature of the constraints, and the length of the predictive horizon [17]. Therefore, it is important to identify formulations which can minimize computational overhead [18], [19].

This paper proposes a unified trajectory planning and control framework for both semiautonomous and autonomous vehicles by outlining a computationally efficient model involving a particle motion description of the target vehicle, the road, and other objects in curvilinear path coordinates. The main module in this framework is referred to as the "predictive trajectory guidance" (PTG) module. This paper details all the components of the nonlinear MPC of the PTG module, including the definitions of constraints involving vehicles/objects, road references, lane limits, traffic rules, such as speed limits, vehicle dynamics limits, and of the suitable set ups of the objective function. Furthermore, two basic interfaces of the PTG are described along with their corresponding lower level vehicle dynamics controllers (VDCs).

A brief version of the approach described in this paper appeared in our conference paper [20]. This paper gives significantly expanded discussions. It details the derivations of the motion and road model (Section III) and the cascade control interfacing to the vehicle dynamics control. It also gives more complete derivations and explanations of the MPC formulation, constraints and objective definitions (Section IV), and configurations for autonomous and semiautonomous driving. We detail the discussion of the parameterization for object distance constraints (Section IV-B2). After introducing the MPC formulation, we remark on the computational burden of solving the optimization problem posed in the MPC. Furthermore, to show the broad applicability of the proposed framework, the results section includes new and updated scenarios compared with [20]. We do defer some details on the lower level vehicle dynamics control options to our prior work [21]. Also, reconfigurations to enable a specific predictive advanced driver assistance mode (called pADAS mode) appeared in [22].

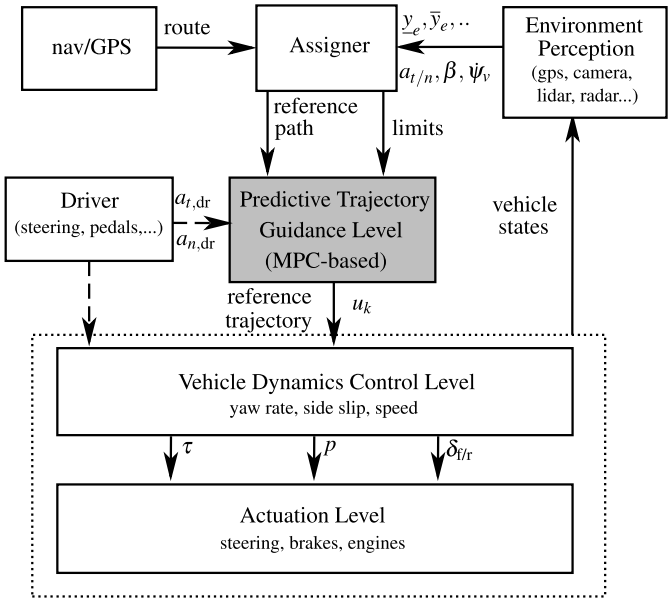

Fig. 1. Multilevel control structure for (semi-)autonomous driving on public roads.

The remainder of this paper is structured as follows. Section II gives the general control framework. Section III describes the particle model used in the PTG level and its relation to the bicycle model commonly used for vehicle dynamics modeling/control. Section IV details the MPC formulation. Section V briefly reviews two options for interfacing the PTG and the vehicle dynamics controllers. Section VI presents results and discussions. Section VII summarizes the conclusions.

II. Control Framework

Fig. 1 shows the multilevel modular framework that can be used to facilitate the discussion of the different interacting functions involved in the control of (semi-)autonomous vehicles. At the top is an assigner module that integrates/fuses the data from the environment perception (cameras, lidar, radar, and so on), route precalculations from navigation/GPS module, vehicle dynamics sensing/estimation modules, and prevailing traffic rules and limits, to select the proper references and constraints for the PTG module. The assigner module can

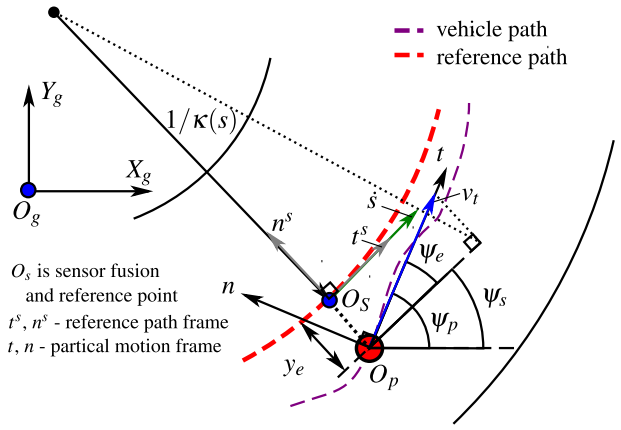

Fig. 2. Definitions of the curvilinear motion description.

be modeled with several finite state machines that set up the optimization problem for the PTG in different maneuvers, such as cruising, following, leading, or changing lanes. Detailed discussions of the assigner module can be found in [23] and [24].

At the PTG level, an MPC algorithm computes the feasible and optimal control inputs for the target vehicle subject to the prevailing constraints and specified objective function. In [21], we identified two options for interfacing the PTG to the VDC level. The first is called a "tracking mode" for the PTG, where its computed optimal input signals (desired longitudinal acceleration and yaw rate of the model used in the MPC) are to be tracked by a feedback plus feed-forward lower level VDC. In the second option, called "planning mode," the PTG merely computes/plans the optimal state trajectories which are then reinterpreted by a trajectory preparation module for a path and speed tracking VDC.

Both autonomous or semiautonomous driving can be realized with the above-mentioned framework [20]. In the autonomous mode, the PTG is considered to track both the reference velocity and lateral position via the accelerator/brake inputs (longitudinal actuation) and active steering (lateral actuation). While in semiautonomous mode, a human driver is responsible for either the longitudinal or the lateral control, while the PTG takes charge of the other.

III. Vehicle Models and Cascade Model Partitioning

This section details the particle model used in the PTG level and its relation to the bicycle model commonly used for vehicle dynamics modeling/control.

A. Particle Motion Model

To detail the model used in the PTG level, we consider the vehicle's motion in relation to a reference path, i.e., a road or lane centerline as shown in Fig. 2. The curvilinear motion of a particle, coincident with the vehicle's center of gravity (CG), is formulated in the Frenet n/t frame with the n- and t- axes normal and tangential, respectively, to its actual path. Introducing a similar frame n^s/t^s for the reference path, the alignment error between the reference frame and the vehicle/particle fixed frame denoted by ψ_e (see Fig. 2), is given by

$$\psi_e = \psi_p - \psi_s \tag{1}$$

where ψ_p gives the vehicle's heading, which is represented by the orientation of the n/t frame attached to the particle/CG with respect to the global frame X_g/Y_g. ψ_s is the rotation of the road frame (labeled with s) with respect to the same global frame. The lateral displacement y_e parallel to the n^s direction locates the particle/CG laterally in relation to the reference path. Its evolution is given by

$$\dot{y}_e = v_t \sin(\psi_e) \tag{2}$$

where v_t is the forward velocity of the particle/vehicle's CG, which is always tangent to the t-axis of the particle motion frame.

Next, consider the particle motion projected on the reference path frame. Using the projection of the particle speed on the reference path frame as shown in Fig. 2, the tangential speed v_t^s and the dynamics of the arc length s read

$$v_t^s = v_t \cos(\psi_e) = (\rho(s) - y_e)\dot{\psi}_s \tag{3a}$$

$$\dot{s} = \frac{ds}{dt} = \rho(s)\dot{\psi}_s \tag{3b}$$

with $\rho(s)$ the distance to the center of instantaneous rotation, which can be written as a function of s. Combining these equations, the arc length dynamics, which can also be viewed as the vehicle speed projected on the reference path, are given by

$$\dot{s} = v_t \cos(\psi_e)\left(\frac{\rho(s)}{\rho(s) - y_e}\right). \tag{4}$$

Then, the rotation of the road frame follows:

$$\dot{\psi}_s = \frac{\dot{s}}{\rho(s)} = v_t \cos(\psi_e)\left(\frac{1}{\rho(s) - y_e}\right). \tag{5}$$

We assume that the road curvature $\kappa(s) = \rho(s)^{-1}$ is known along the arc length s. We also find that using road curvature instead of the road radius has at least two advantages in our modeling. First, straight roads are dealt with readily by setting the curvature $\kappa(s) = 0$, while the radius would go to infinity and pose numerical difficulties. Second, many roads would involve continuous evolution of finite curvature values that change signs, while using radius would involve jumps, e.g., $+\infty$ and goes to $-\infty$. Furthermore, we approximate available road curvature information in fitted-polynomial form

$$\kappa(s) = c_{\kappa,0} + c_{\kappa,1}s + c_{\kappa,2}s^2 + c_{\kappa,3}s^3 + \cdots \tag{6}$$

where $c_{\kappa,i}, i = 0, 1, 2, \ldots$ are the polynomial coefficients. These coefficients are assumed to be mapped to the real road curvature by a suitable algorithm within the environmental perception module. Typically, a high-order polynomial (≥ 10) may be needed to sufficiently describe the reference path for a selected predictive horizon. Other feasible ways to describe the reference path include using combinations of Gaussian or hyperbolic functions, especially for describing the combinations of straight roads and sharp corners as usually found in urban environments.

Introducing a_t for the longitudinal acceleration of the vehicle's CG (in particle description), the final model for the curvilinear particle motion description is given by

$$\dot{v}_t = a_t \tag{7a}$$
$$\dot{\psi}_e = \dot{\psi}_p - v_t \cos(\psi_e)\left(\frac{\kappa(s)}{1 - y_e\kappa(s)}\right) \tag{7b}$$
$$\dot{y}_e = v_t \sin(\psi_e) \tag{7c}$$
$$\dot{s} = v_t \cos(\psi_e)\left(\frac{1}{1 - y_e\kappa(s)}\right). \tag{7d}$$

In addition, assuming that the (lower level) closed-loop vehicle dynamics exhibits a first-order lag behavior, the generation of a_t and $\dot{\psi}_p$ by the controlled vehicle can be approximated by the first-order dynamics with time constants T_{a_t} and $T_{\dot{\psi}_p}$ as follows:

$$\dot{a}_t = 1/T_{a_t}(a_{t,d} - a_t) \tag{8a}$$
$$\ddot{\psi}_p = 1/T_{\dot{\psi}_p}(\dot{\psi}_{p,d} - \dot{\psi}_p). \tag{8b}$$

Note that the time constants T_{a_t} and $T_{\dot{\psi}_p}$ are the only parameters in the model described earlier that are related to the controlled vehicle. The masking of this first-order closed-loop controlled dynamics in the overall cascade control setup is shown in Fig. 5. It is also possible to introduce a higher order dynamics model for the controlled vehicle to replace this first-order approximation at the cost of increased computational burden.

Remark 1: From (8a) and (8b), the desired longitudinal acceleration $a_{t,d}$ and the desired yaw rate $\dot{\psi}_{p,d}$ are considered as the control inputs to the entire model described thus far. However, given the quadratic objective functions we shall pose later for the MPC in the predictive guidance module, it is often desired to minimize the control input. However, this contradicts, for example, path tracking objectives in steady turning, which require $\dot{\psi}_p = a_n/v_t \neq 0$, where a_n is the lateral/normal acceleration. It is, therefore, convenient to introduce a new control input $\Delta\dot{\psi}_{p,d}$, which describes the deviation from a nominal yaw rate $\dot{\psi}_{p,r}$ necessary to follow the reference path at the current speed. Formally, we write

$$\dot{\psi}_{p,d} = \dot{\psi}_{p,r} + \Delta\dot{\psi}_{p,d} \tag{9a}$$
$$= v_t\kappa(s) + \Delta\dot{\psi}_{p,d}. \tag{9b}$$

In fact, by assuming a free road and no external disturbances, the particle can follow the given reference path $\kappa(s)$ under $\dot{\psi}_{p,r}$ with new control input $\Delta\dot{\psi}_{p,d} = 0$, which is consistent with an input minimization objective in the MPC.

Substituting (9b) into (8b), we note that the final inputs to the motion model are $a_{t,d}$ and $\Delta\dot{\psi}_{p,d}$.

B. Relationships Between Particle Motion Description and Bicycle Model

1) Bicycle Model: The model often used at the VDC level to describe the dynamic behavior of a road vehicle is the bicycle (or single track) model shown in Fig. 3. The corresponding

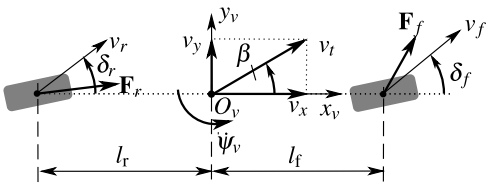

Fig. 3. Definitions in the bicycle (single track) model.

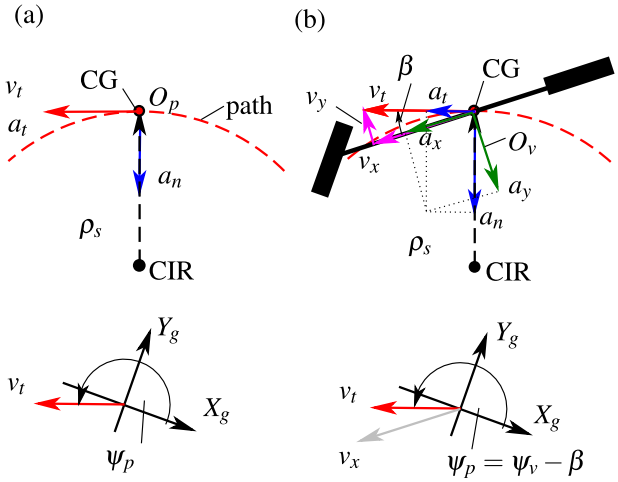

Fig. 4. (a) Kinematics of particle motion and (b) vehicle motion variables and the mapping between them.

equations of motion are

$$\dot{v}_t = \frac{1}{m}\left(\sum_{q=f,r} F_{x,q}\cos\beta + \sum_{q=f,r} F_{y,q}\sin\beta\right) \tag{10a}$$
$$\dot{\beta} = \frac{1}{mv_t}\left(-\sum_{q=f,r} F_{y,q}\sin\beta + \sum_{q=f,r} F_{x,q}\cos\beta\right) - \dot{\psi}_v \tag{10b}$$
$$\ddot{\psi}_v = \frac{1}{J_z}\left(l_f F_{y,f} - l_r F_{y,r} + M_{a,f} + M_{a,r}\right) \tag{10c}$$

where β is the side-slip angle of the vehicle, ψ_v is the yaw angle of the vehicle in the global frame, m is the vehicle mass, and J_z is the yaw inertia. $F_{x/y,q}$ and $M_{a,f/r}$ are the tire forces and the aligning moments resolved in the vehicle fixed frame O_v. The subscript q represents either front f or rear r. The tire forces and the aligning moments are generally nonlinear functions of vehicle states and actuator inputs, such as steering angle and wheel torque. The interested reader is referred to the common literature in this field for the details [25]–[27].

2) State and Control Mapping Between Particle Motion Description and Bicycle Model: The mapping discussed in this section is relevant for properly interfacing the particle motion model to be used at the PTG level and the bicycle model to be used at the VDC level. In Fig. 4(a), from planar particle motion kinematics, the acceleration has the tangential a_t and the normal acceleration a_n components given by

$$\begin{bmatrix} a_t \\ a_n \end{bmatrix} = \begin{bmatrix} \dot{v}_t \\ \dfrac{v_t^2}{\rho_s} \end{bmatrix} \tag{11}$$

with ρ_s representing the distance to the instantaneous center of the actual vehicle path. Note that the angular deviation between the particle attached frame O_p and the vehicle attached (body-fixed) frame O_v is the side-slip angle β, as shown in Fig. 4(b). Therefore, we have

$$\psi_p = \psi_v - \beta. \tag{12}$$

From kinematics, the centrifugal acceleration is determined by its rotational speed and the distance to the center of rotation. Thus, for the particle and the vehicle's CG (instantaneously coincident), we have

$$\dot{\psi}_p = \frac{v_t}{\rho_s} = \dot{\psi}_v - \dot{\beta} \tag{13}$$

and thus

$$a_n = \frac{v_t^2}{\rho_s} = v_t \dot{\psi}_p = v_t(\dot{\psi}_v - \dot{\beta}). \tag{14}$$

This eliminates the need for knowing ρ_s, which is hard to determine in (11). Instead, using vehicle states $\dot{\psi}_v$ and $\dot{\beta}$, one can determine the accelerations for the particle motion model from the vehicle dynamics. This approach is used in the closed-loop simulation results to be presented later. Also, one can easily show that the relations between the vehicle's acceleration and velocity at the CG resolved in the vehicle attached frame O_v and that of the particle motion frame O_p are given by

$$\begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} a_t \cos\beta + a_n \sin\beta \\ a_t \sin\beta - a_n \cos\beta \end{bmatrix} \tag{15a}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v_t \cos\beta \\ v_t \sin\beta \end{bmatrix}. \tag{15b}$$

Remark 2: As different models are used in the PTG and VDC levels, a transformation of the particle motion control inputs $a_{t,d}$ and $\Delta\dot{\psi}_{p,d}$ computed in the PTG is needed to meet the specifications of the control systems in VDC. For example, these control systems usually use the vehicle yaw rate ($\dot{\psi}_v$) as control reference to manipulate the motion of the vehicle and its directional stability [28], [29]. Therefore, using (9) and (13), we have

$$\dot{\psi}_{v,d} = \dot{\psi}_{p,d} + \dot{\beta} = v_t\kappa(s) + \Delta\dot{\psi}_{p,d} + \dot{\beta} \tag{16}$$

where measured or estimated information is assumed to be available about the side-slip angle β and its derivative. In addition, for acceleration control, the particle motion control inputs $a_{t,d}$ and $\Delta\dot{\psi}_{p,d}$ can be transformed to the vehicle attached frame via (9), (14), and (15a)

$$\begin{bmatrix} a_{x,d} \\ a_{y,d} \end{bmatrix} = \begin{bmatrix} a_{t,d}\cos\beta - v_t(v_t\kappa(s) + \Delta\dot{\psi}_{p,d})\sin\beta \\ a_{t,d}\sin\beta + v_t(v_t\kappa(s) + \Delta\dot{\psi}_{p,d})\cos\beta \end{bmatrix}. \tag{17}$$

Either these or the desired acceleration $[a_{t,d}\ \ a_{n,d}]^T$ (with proper transformations, $a_{n,d} = v_t\Delta\dot{\psi}_{p,d}$) can be used as control references for the control systems at the VDC level. If the side-slip angle information is not available, the assumptions $\beta \approx 0$ and $\dot{\beta} \approx 0$ can be made for some maneuvers.

Finally, we remark that it is also possible to establish similar relationships between the particle motion model and other higher fidelity vehicle dynamics models, should one need to replace the bicycle model at the VDC level.

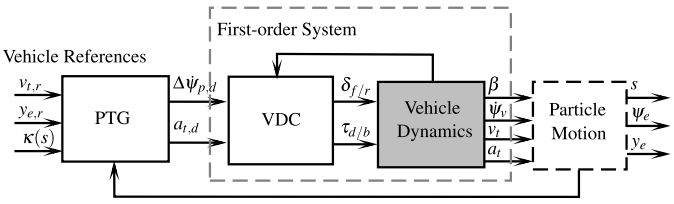

Fig. 5. Cascade control setup.

C. Cascade Control Setup

Fig. 5 shows the cascade control setup that interfaces the PTG and VDC levels. Herein, in the PTG level, vehicle references, such as desired velocity $v_{t,r}$, desired lateral position $y_{e,r}$, and curvature of the reference path $\kappa(s)$, are compared with the particle motion states which are interpreted from the vehicle dynamics states according to the discussion in the Section III.B. The tracking errors are used by the PTG module to generate the desired control inputs $a_{t,d}$ and $\Delta\dot{\psi}_{p,d}$ for the particle motion model. These inputs are then sent to and interpreted by the control systems at VDC level to generate appropriate actuation to the vehicle. In this particular setup of Fig. 5, the VDC uses the front and rear steering angles δ_f and δ_r, the rear wheel drive torque τ_d, and the brake torques $\tau_{b,f/r}$ to control the vehicle's dynamics. The application of rear steering can improve the agility and stability in tracking the reference from the PTG level. In the Sections IV and V, the PTG and VDC levels will be discussed in detail.

IV. Predictive Trajectory Guidance Level: MPC Formulation

A. Traffic, Lane, and Vehicle Dynamics Limits

We first express the lane limits and traffic restrictions, such as speed limits and stop signs, using a similar approach adopted for the definition of the curvature of the reference path (road/lane centerline) by (6). Many of these limits are generally functions of the spatial position as the independent variable, while the model outlined earlier is posed with time as the independent variable. The introduction of the arc length s as one of the particle motion state variables [see (7d)] simplifies the statement of the relevant constraints thereby overcoming a potential problem with mixed independent variables. For example, upper speed limits $\overline{v}_t(s)$ as well as the upper and lower lateral position/lane limits $\overline{y}_e(s)$, $\underline{y}_e(s)$ can be expressed as

$$\overline{v}_t(s) = c_{0,v} + c_{1,v}s(t) + c_{2,v}s(t)^2 + c_{3,v}s(t)^3 + \cdots \tag{18a}$$

$$\overline{y}_e(s) = c_{0,ll} + c_{1,ll}s(t) + c_{2,ll}s(t)^2 + c_{3,ll}s(t)^3 + \cdots \tag{18b}$$

$$\underline{y}_e(s) = c_{0,lr} + c_{1,lr}s(t) + c_{2,lr}s(t)^2 + c_{3,lr}s(t)^3 + \cdots \tag{18c}$$

$$s(t) \le \overline{s}(t) \tag{19}$$

where $c_{i,v}, c_{i,ll}, and c_{i,lr}, i = 0, 1, 2, \ldots$ are the polynomial coefficients may be computed by the assigner module (Fig. 1) using information of the environmental sensors and traffic restrictions. $\overline{s}(t)$ is used to limit the longitudinal position of the autonomously controlled vehicle (ACV) such as necessary to enforce a stop sign or red traffic light. It should also be noted that the definitions of $\overline{y}_e(s)$ and $\underline{y}_e(s)$ are relative to

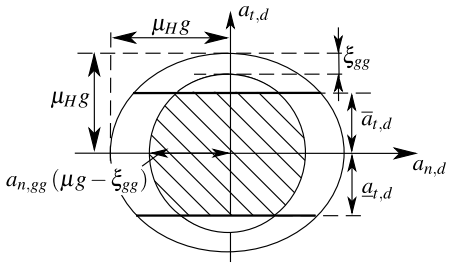

Fig. 6. Definitions of acceleration limits.

the reference path and not in the absolute/inertial frame O_g. Then, straightforward inequality constraints can be added for the speed and lateral position of the vehicle as

$$\Delta\overline{v}_t = \overline{v}_t(s) - v_t \geq 0 \tag{20a}$$
$$\Delta\overline{y}_e = \overline{y}_e(s) - y_e \geq 0 \tag{20b}$$
$$\Delta\underline{y}_e = y_e - \underline{y}_e(s) \geq 0. \tag{20c}$$

Overbars and underbars indicate the upper and lower limits, respectively.

We also apply the well-known tire-road friction ellipse, which constrain the longitudinal and lateral accelerations, as shown in Fig. 6

$$(\mu_H g - \zeta_{gg})^2 \geq \left(\frac{a_{n,d}}{a_{n,gg}}\right)^2 + (a_{t,d})^2 \tag{21a}$$
$$a_{n,d} = v_t(\kappa(s)v_t + \Delta\dot{\psi}_{p,d}) \tag{21b}$$
$$0 \leq \zeta_{gg} \leq \overline{\zeta}_{gg} \tag{21c}$$
$$0 \leq s \leq \overline{s}. \tag{21d}$$

Herein, the first constraint in (21a) is intended to restrict the input accelerations $a_{t,d}$ and $a_{n,d}$ [$a_{n,d}$ is derived from (9) and (14)] to the physical ability of the tire/road contact patch. This is represented by the elliptical region with half minor/major axes $\mu_H g - \zeta_{gg}$ and $a_{n,gg}(\mu_H g - \zeta_{gg})$ in Fig. 6. μ_H is the maximum friction coefficient of the tire/road contact patch. The normalization coefficient $a_{n,gg} \leq 1$ provides a parameterization for tuning the lateral acceleration limit. The slack variable ζ_{gg} and its upper limit $\overline{\zeta}_{gg}$ were introduced in [20]. These make it possible to lower the vehicle accelerations for the sake of more comfortable trajectories except in critical maneuvers, where the slack variable allows full use of the acceleration potential. The slack variable ζ_{gg} is formulated to track a desired value $\zeta_{gg,r}$ and the two norm of the tracking error $\zeta_{gg} - \zeta_{gg,r}$ is weighted in the objective function of the MPC in the PTG level. This formulation is often called a soft constraint. Furthermore, the constraints

$$\underline{a}_{t,d} \leq a_{t,d} \leq \overline{a}_{t,d} \tag{22a}$$
$$\underline{a}_{n,d} \leq (v_t\kappa(s) + \Delta\dot{\psi}_p)v_t \leq \overline{a}_{n,d} \tag{22b}$$

limit the control inputs to the physical potential of the vehicle from actuation perspectives by adding limits for braking $\underline{a}_{t,d}$, traction $\overline{a}_{t,d}$, and steering $\underline{a}_{n,d}, \overline{a}_{n,d}$. The hatched area in Fig. 6 shows the final available acceleration input space (hard constraints on $a_{n,d}$ are omitted for clarity).

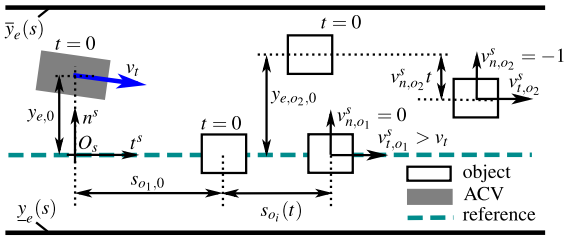

Fig. 7. Object motion definition ($\kappa = 0$) in road reference frame.

B. Spatial Constraints for Other Objects on the Road

1) Definitions of Objects for CA: One method of defining other objects to avoid is to use potential fields as used in a previous work [30]. However, that approach will allow the vehicle to crash into an obstacle if the weightings in the objective function are not selected properly. A proper weighting is hard to find in the presence of multiple objects and other competing terms in the cost function. Therefore, in this paper, we outline a new approach, where each object moves along the curvilinear frame of the reference path with its own longitudinal and lateral speeds and accelerations. This information is assumed to be obtained by the top-level assigner module from vehicle-to-vehicle communication. During an MPC prediction horizon (between updates), we assume that all objects follow their path with the constant longitudinal and lateral accelerations a^s_{t,o_i} and a^s_{n,o_i}, starting at their position $s_{o_i,0}$ and $y_{e,o_i,0}$, and speed v^s_{t,o_i} and v^s_{n,o_i} at the beginning of the prediction. These speeds and accelerations are defined in the path curvilinear frame. With this assumption, the arc length s_{o_i} and the lateral distance y_{e,o_i} to the reference line (the center of the road lane) for the ith object are given by (see Fig. 7)

$$s_{o_i} = s_{o_i,0} + v^s_{t,o_i}x_t + \frac{1}{2}a^s_{t,o_i}x_t^2 \tag{23a}$$
$$y_{e,o_i} = y_{e,o_i,0} + v^s_{n,o_i}x_t + \frac{1}{2}a^s_{n,o_i}x_t^2 \tag{23b}$$
$$\dot{x}_t = 1. \tag{23c}$$

The state x_t is added in order to trace the MPC internal time. At each MPC update, the initial value of x_t will be set to zero. This allows for calculating the positions of the objects in the prediction horizon using only the measurement of $v^s_{t,o_i}, v^s_{n,o_i}, a^s_{t,o_i}$, and a^s_{n,o_i}. This is done without introducing additional motion states for each object as we did in a previous work [30]. Herein, no heading error of the objects is assumed. This is equivalent to the practical assumption that the object recognition module can project the object speeds to the road frame. With these object positions computed over the prediction horizon, elliptical constraints can be used to restrict the ACV from getting close to and colliding with any object

$$1 \leq \left(\frac{y_e - y_{e,o_i}}{\Delta y_{e,o_i}}\right)^2 + \left(\frac{s - s_{o_i}}{\Delta s_{o_i}}\right)^2. \tag{24}$$

Herein, $\Delta y_{e,o_i}$ is the lateral width plus clearance of the object i, and thus, it is the distance of the ACV to the object when passing; and Δs_{o_i} is the distance and clearance to the vehicle along the arc length. It should be noted that the

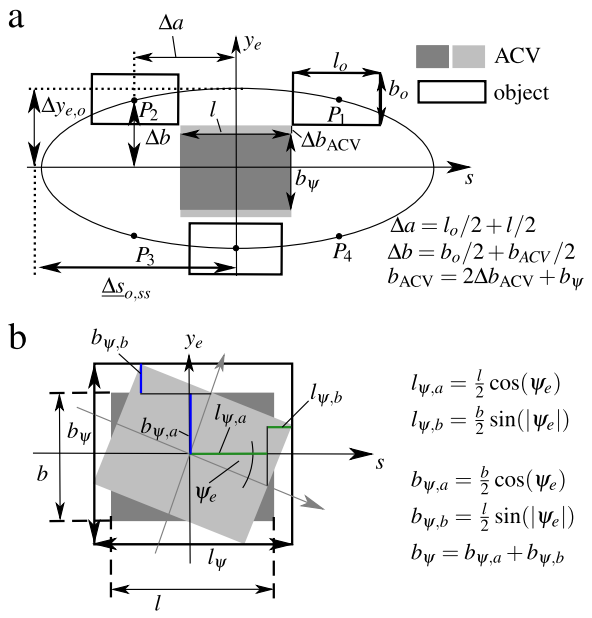

Fig. 8. Distance constraint parameter definition for $f_{\zeta,Do} = 0$.

distance Δs_{o_i} is not a fixed value as the speed of the ACV is not constant over the prediction horizon and traffic rules could impose a speed-dependent minimum distance. Thus, distances to objects and the distance slack limit are represented by

$$\Delta s_{o_i} = \Delta s_{o,ss} + f_{\zeta,Do}\zeta_{Do} \tag{25a}$$

$$0 \leq \zeta_{Do}. \tag{25b}$$

Herein, $\Delta s_{o,ss}$ is a static portion of distance to objects (see Section IV-B2 for more information) and $f_{\zeta,Do}\zeta_{Do}$ is the dynamic portion of distance to objects defined by a free parameter $f_{\zeta,Do}$ and the slack variable ζ_{Do}, which was introduced in [20]. $f_{\zeta,Do}$ gives an optional parameter to manipulate the behavior of the trajectory guidance algorithm, for example, by setting $0 \leq f_{\zeta,Do} \leq 2$. The slack variable ζ_{Do} is used to simulate a soft constraint that allows the PTG to manipulate the longitudinal distance to obstacles in critical maneuvers such as CA.

It should be noted that the definition of the elliptical constraint as mentioned earlier may lead to two equally possible homotopy class of possible trajectories, as when the ACV is heading along the major axis of the elliptic constraint: either passing through the left or the right side of the ellipse. In these cases, the solution of optimization problem will depend on the initial guess of the ACV state. A preference to one side (say, passing in the left by driving custom) can be embedded in the optimization by adding a very small offset in the reference trajectory to be tracked. Unless stated otherwise, we adopt the later approach in the simulation results presented later.

2) Parameterization of Elliptical Constraints for Avoidance of Objects: The parameters of the elliptic avoidance constraints between objects and the ACV are related to the physical (geometric) size of the objects. Assuming that the environment recognition module measures/estimates the sizes of objects, the parameters of the ellipse are calculated according to Fig. 8(a). First, the critical situation is depicted in the figure with the ACV touching the corner of an object [see P_1, P_2, P_3, P_4, and their positions in Fig. 8(a)]. The task is then to find the safe half minor ($\Delta y_{e,o}$) and the half major ($\Delta s_{o,ss}$) axes of the ellipse. The geometric size of the ACV is defined by its length l and its width b, while each object has its own length l_o and width b_o. A small margin Δb_{ACV} is added to the lateral distance between the ACV and objects to ensure safety. Using constructions like Fig. 8 and taking an arbitrary point $(s_i, y_{e,i})$ on the ellipse, the following relation can be written:

$$\left(\frac{s_i}{\Delta s_{o,ss}}\right)^2 + \left(\frac{y_{e,i}}{\Delta y_{e,o}}\right)^2 = 1. \tag{26}$$

Substituting any one of the critical points P_1, P_2, P_3, and P_4 in Fig. 8(a) (which mirrors the object around the ACV with the position deviations of Δa and Δb in particle frame) to (26), we obtain

$$\left(\frac{\Delta a}{\Delta s_{o,ss}}\right)^2 + \left(\frac{\Delta b}{\Delta y_{e,o}}\right)^2 = 1. \tag{27}$$

To solve for $\Delta y_{e,o}$ and $\Delta s_{o,ss}$ from (27), infinite combinations of solutions can be found. Considering the unified scale of the ellipse according to the position of critical point P_1, P_2, P_3, and P_4. The following solution is adopted:

$$\Delta s_{o,ss} = \sqrt{2\Delta a^2} \tag{28a}$$

$$\Delta y_{e,o} = \sqrt{2\Delta b^2}. \tag{28b}$$

The introduced elliptic constraint is still not detailed enough to guarantee a safe distance to objects. In fact, when the ACV changes a lane, it does not move tangentially to the road reference anymore, and thus, its original dimensions for straight driving b, l will change, see Fig. 8(b). In this case, the size of the ACV is projected onto the road reference frame O_s according to

$$l_\psi = l_{\psi,a} + l_{\psi,b} = \frac{l}{2}\cos(\psi_e) + \frac{b}{2}\sin(|\psi_e|) \tag{29a}$$

$$b_\psi = s_{\psi,a} + s_{\psi,b} = \frac{b}{2}\cos(\psi_e) + \frac{l}{2}\sin(|\psi_e|). \tag{29b}$$

under the assumption that $\psi_e < \pi/2$. Fig. 8 shows the detail kinematic relations.

C. Constraints to Circumvent Model Singularities

The introduced curvilinear particle motion description includes a singularity that could make it hard to use for the optimizations in MPC. Specifically, the reformulation of the aligning error with the road curvature κ results in a singularity in (7) for $y_e\kappa = 1$. Assuming a lateral error y_e in the range of 1 m, a value of $\kappa < 1$ is allowed, which is a narrow circle of a radius 1 m. As this is not common for public road designs, this singularity is of theoretical nature only. Nevertheless, to ensure numerical stability, we add the inequality constraint

$$y_e\kappa < 1. \tag{30}$$

Another constraint affecting the curvature is imposed by the turning radius of the actual vehicle. From the kinematics of vehicle motion and the limits of the steering angle $\overline{\delta}_f$, the

vehicle wheel base l_{wb}, and track width b_{tr}, the curvature limit for a front-steered vehicle is given by [26]

$$\overline{\kappa} = \frac{1}{\frac{b_{tr}}{2} + \frac{l_{wb}}{\sin \overline{\delta}_f}}. \tag{31}$$

This would be used to put a constraint on the yaw rate input to be computed from (9) as

$$\psi_{p,d} = v_t \kappa + \Delta \dot{\psi}_{p,d} \leq v_t \overline{\kappa}. \tag{32}$$

D. MPC Formulation

We now outline the nonlinear MPC scheme for the PTG module using the models and constraints detailed so far. To this end, we first recall and define some MPC related notations. The MPC implementation we outline solves a discretized optimization problem over a receding prediction horizon. We denote each discretization step by k for $k \in (1, 2, \ldots, N_p)$ with the prediction horizon length N_p and sampling time step ΔT of the model discretization, where $N_p = H_p/\Delta T$. Suppose the prediction is started at the absolute time t_0, at prediction step k, the state x_k denotes the predicted motion at absolute time $t = t_0 + k\Delta T$. As our version of the MPC solver does not allow a separate slack in the cost, the two slack variables mentioned earlier are introduced as additional model states with the auxiliary dynamics

$$\dot{\zeta}_{gg} = u_{\zeta,gg} \tag{33a}$$

$$\dot{\zeta}_{Do} = u_{\zeta,Do}. \tag{33b}$$

Therefore, two additional inputs need to be added for the MPC optimization to accommodate these states. Note that, while it is possible to treat the two slack variables as inputs in the optimization, we opt to treat them as states with allowed dynamic change between discretization steps.

The finite horizon optimization problem to be solved in the MPC is the following:

$$\min_{u_1,\ldots,u_{N_p-1}} \sum_{k=1}^{N_p} \underbrace{||y_k - r_k||^2_Q}_{\text{reference tracking}} + \sum_{k=0}^{N_p-1} \underbrace{||u_k - u_{r,k}||^2_R}_{\text{control minimization}} \tag{34}$$

subject to

$$\dot{x} = f(x, u) \tag{35a}$$

$$x(t_0) = x_0 \tag{35b}$$

$$x_k = x(t_0 + k\Delta T) \tag{35c}$$

$$y_k = h(x_k) \tag{35d}$$

$$r_k = r(t_0 + k\Delta T) \tag{35e}$$

$$u_k = u(t_0 + k\Delta T) \tag{35f}$$

$$u_{r,k} = u_r(t_0 + k\Delta T) \tag{35g}$$

$$c(x, u) \leq 0 \tag{36}$$

where

$$x = [v_t \quad y_e \quad \psi_e \quad s \quad x_t \quad a_t \quad \dot{\psi}_p \quad \zeta_{gg} \quad \zeta_{Do}]^T \tag{37a}$$

$$y = [y_e \quad v_t \quad \zeta_{gg} \quad e_{\zeta,Do}]^T \tag{37b}$$

$$r = [y_{e,r} \quad v_{t,r} \quad \zeta_{gg,r} \quad e_{\zeta,Do,r}]^T \tag{37c}$$

$$u = [a_{t,d} \quad \Delta \dot{\psi}_{p,d} \quad u_{\zeta,gg} \quad u_{\zeta,Do}]^T \tag{37d}$$

$$u_r = [a_{t,d,r} \quad \Delta \dot{\psi}_{p,d,r} \quad u_{\zeta,gg,r} \quad u_{\zeta,Do,r}]^T. \tag{37e}$$

The objective function by (34) penalizes the summation of the equidistantly sampled [see (35c)–(35g)] reference tracking and control errors. Q and R are appropriate positive semidefine weighting matrices. The augmented state equations (35a), with the augmented state vector given by (37a), are assumed discretized with sample time ΔT with piecewise constant inputs u at prediction steps k. The problem is then to minimize the objective function subject to the equality constraints in (35) and inequality constraints compactly represented by (36).

For reference tracking, the relevant errors to minimize are those between the system outputs in y [(37b)] and their respective references in r[(37c)]. The system outputs include the speed v_t and lateral position y_e of the particle model and the two slack variables. The references for the speed $v_{t,r}$ and lateral position $y_{e,r}$ are specified by the assigner module according to the environmental and route/navigation information. The reference for the slack variable ζ_{gg} is selected near the upper bound $\overline{\zeta}_{gg}$, which directly defines the reduction of the vehicle acceleration/handling limit [see (21a)]. For the last output component $e_{\zeta,Do}$, which is defined by $e_{\zeta,Do} = v_t - \zeta_{Do}$, the reference is simply set to a zero value. With this selection, the distance slack variable is made automatically speed dependent for the prediction horizon.

By including a control error term in the objective function as the deviation of the control efforts in u from their desired references u_r, it is sought to tradeoff the driving characteristics (e.g., comfort, sport, and safety) as well as to model semiautonomous and fully autonomous driving modes in a unified fashion. The control efforts in u incorporate the inputs to the motion model, namely, $a_{t,d}$, $\Delta \dot{\psi}_{p,d}$ (see *Remark 1*), and those to the auxiliary dynamics of the slack variables, namely, $u_{\zeta,gg}$ and $u_{\zeta,Do}$ [see (33)]. The references of the control inputs may be configured differently to model distinct operation modes. For example, in fully autonomous driving, all components in u_r are set to zero to minimize the control efforts. However, in semiautonomous driving, the driver inputs can be interpreted as some of the control references.

The inequality constraint set in (36) covers the aforementioned constraints: the traffic and physical limits in (19)–(22), the CA constraints with any objects defined in (24) and (25b), and the constraints for avoiding singularity described in (30) as well as the curvature constraint in (32).

Remark 3: As already noted following (23), by adding the prediction time x_t as a state of the prediction model (37a), we reduce the model order compared with the approach in [30], where the dynamics of each obstacle object were added as additional states for the model in the MPC. The object positions are now obtained algebraically from (23).

Remark 4: The choice of the predictive horizon length N_p and sampling step ΔT is a tradeoff between optimality and the problem size for the optimization/computational burden. In the present application, $N_p = 40$ and $\Delta T = 0.15$ gave a good compromise covering all tested traffic scenarios with the particular software implementation described in Section IV-F. More details about tuning procedures for MPC can be found in [31] and [32].

Remark 5: While the discretization of the model for MPC uses the sample time $\Delta T = H_p/N_p$, which is selected as a

value suitable for the application at hand, the MPC update interval T_{mpc} (sampling time for the guidance module) can be selected separately considering the possible execution time (solver algorithm and hardware) for real-time implementation. There is a tradeoff between a desire for fast MPC update (small T_{mpc}) and a fast sampling time for a given prediction horizon. To be more precise, short T_{mpc} allows the MPC-based guidance module to use the latest information and be more adaptive to environmental changes as could be needed in complex traffic. However, short ΔT requires more sample intervals for a given desired prediction horizon, which also increases the problem size for the optimization problem to be solved at each MPC update, and thereby, the execution time. Therefore, in practice, T_{mpc} is often selected to be smaller than ΔT, which is called *oversampling* [33].

E. Fully Autonomous and Semiautonomous Modes

As already mentioned, the selection of the control references in the objective function is one way to prioritize different fully autonomous and semiautonomous driving objectives. Another way is the selection of the weighting matrices Q and R in the objective function (34).

For the fully autonomous driving, the objective of following the middle of a lane $y_{e,r}$ is combined with tracking a reference velocity $v_{t,r}$ using both the longitudinal and lateral control inputs. The reference velocity is obtained from the assigner module according to the perception of the speed limits or the passengers' preference [23], [24]. Sometimes a deviation $y_e - y_{e,r} \neq 0$ may appear as an optimal tradeoff between speed and path tracking, accelerations (control), and other constraints. The slack weights $Q_{sl,gg/Do}$ are set high to generate comfortable trajectories by restricting the use of full acceleration potential except in safety-related situations, e.g., for CA. In those situations, the soft limit defined by ζ_{gg} and the distance to objects via ζ_{Do} are reduced automatically to generate feasible solutions in the presence of hard constraints like the distance to obstacle object $s - s_{o_i}$.

Semiautonomous driving constitutes various advanced driver assistance system functions that are possible to incorporate in the control framework shown in Fig. 1. Therein, parts of controlling the semi-ACV is conducted by the driver. Here, the hard limits for the control inputs in (22) are chosen differently from the fully autonomous mode. For example, for adaptive cruise control (ACC), the longitudinal control reference $a_{t,d,r}$ may be set to zero and the MPC in the PTG executes all longitudinal control; while for lane keeping assist (LKA) or CA, the lateral control references $\Delta\dot{\psi}_{p,d,r}$ come from driver steering intent interpretation modules with the longitudinal speed controlled by a human driver.

In Section VI, some results are included to illustrate these functionalities of the unified control framework for both autonomous and semiautonomous driving.

F. Software Implementation and Computational Complexity

The MPC formulated above is implemented using the ACADO Toolkit and accompanying Code Generation Tool from [9], [10], and [34]. Therein, a sequential-quadratic programming algorithm generates a quadratic approximation of the nonlinear problem and solves it with an online active set QP solver from the open-source library qpOASES. Some restrictions and facilities in the tool influenced the formulation of the MPC model adopted here more than others. For example, instead of formulating the obstacles to avoid as polygons, each of which are in turn defined by intersections and unions of regions defined by linear constraints, the elliptic constraint formulation allows one to use few analytical functions to describe each obstacle. The drawback is this could lead to some conservatism in describing the avoidance area. In our analysis, we used the multiple shooting horizon discretization option and a fourth-order explicit Runge–Kutta integrator. For details on the toolbox, the interested reader is referred to [34] and the references therein.

While it is not possible to give the upper bounds on the number of iterations needed to arrive at the optimal solution of the nonlinear optimization problem with active set QP solvers, some computational scalability bounds can be given for a single QP iteration. Following discussions in [17], it can be shown that the computational complexity of one QP iteration of the active set solver selected is $O(N_x^3 + N_u^2 + N_p^2 + (N_u N_p)^2 + N_u N_p N_c)$, where N_x is the number of states, N_u is the number of control inputs, N_p is the length of the prediction horizon, and N_c is the number of constraints. Thus, the additional control inputs, states, and constraints we added for the slack variables lead to an increase of the computational load. However, these additions are deemed acceptable as they make the problem more tractable.

V. Vehicle Dynamics Control Level

As briefly mentioned in the introduction of the overall control framework (with Fig. 1), the VDC can have two configurations: tracking mode and planning mode. Here, we only give a shortened overview of the VDC in each mode and refer the reader to [21] for details.

A. Tracking Mode

In this mode, the longitudinal control and lateral control inputs computed by the MPC at the PTG level (or from the interpreted driver inputs in semiautonomous mode) are passed to as references to traditional lower level VDCs. For longitudinal tracking, a feed-forward and a PID-feedback control are used to compute the necessary wheel torque, which is saturated to physical limits and then assigned to the engine or brakes (driving or braking torque $\tau_{d/b}$). The PID-feedback part uses the longitudinal acceleration error between the actual a_t and desire ones $a_{t,d}$, while the feed-forward part directly uses the PTG generated reference $a_{t,d}$ to compute the required torque $\tau_{d/b}$. For lateral tracking, a speed-dependent Proportional Integral Derivative (PID)-feedback controller computes the desired steering angle $\delta_{f,d}$ of the front axle steering actuator based on the desired yaw rate of the vehicle $\psi_{v,d}$. The steering angle is saturated when a given side-slip angle of the front tires is reached. The side-slip angle can be estimated via various observer designs [35]. Also, a rear steering actuator may be added to improve agility and stability in tracking the reference generated at the PTG level.

B. Planning Mode

The key idea in the planning mode is to take advantage of the information about the collision-free vehicle state trajectory computed by the PTG level. A *trajectory preparation* module interprets the computed trajectory from MPC for the prediction horizon along with the vehicle position information and outputs reference trajectories for the lower level speed and path tracking controller. Details are given in [21]. The particular speed and path tracking controller we adopted is derived via an input–output linearization of the bicycle model dynamics with a front-decoupling point. It uses cascaded position and speed control loops to align the front-decoupling point to the desired point on the reference trajectory as detailed in [36].

C. Comparison of Tracking Mode and Planning Mode

For autonomous driving, our experiments indicate that both modes can give satisfactory performance in simple maneuvers such as cornering with lateral acceleration $a_n \leq 4.5$ m/s^2. However, for aggressive maneuvers, such as a race line scenario featuring several sharp corners, significant reference tracking errors and violations of both acceleration and road constraints appear when using the planning mode, but not the tracking mode. This is not unexpected since the PTG merely plans the trajectory in the planning mode and the low level control is decoupled from the constraints used in the MPC in this mode. Readers interested in further details on the distinction and performance of the two modes are referred to [21]. Here, for the simulation results that follow, we choose the more reliable tracking mode at the VDC level as the main configuration of the proposed framework.

VI. Results and Discussion

In this section, several cases are simulated to illustrate the performance of the proposed control framework in both autonomous and semiautonomous driving. A high-fidelity two-track vehicle model that includes nonlinear tires, tire force relaxation, load transfer, driving resistances, and actuator dynamics is used for representing the controlled vehicle [26], while the PTG uses the motion model and MPC formulation detailed in the Section IV.D implemented with the ACADO code generated solver described earlier. The controllers at the VDC level are sampled with 1 ms commensurate with the fast lower level vehicle dynamics. Unless specified otherwise, the following MPC settings are used: the prediction model is discretized with $\Delta T = 0.15$ s and $N_p = 40$, and thus $H_p = 6$ s the MPC is updated with $T_{mpc} = 0.05$ s (this is higher than all execution times tested). The other parameters chosen are listed in Table I.

TABLE I
Parameter Selection for Different PTG Modes

Driving Mode	Semi-Autonomous		Fully Autonomous
	ACC	LKA & CA	
Q_{y_e}	3		2
Q_{v_t}	1.1		1.1
Q_{sl}	40		20
R_{a_t}	20		20
$R_{\Delta\psi_v}$	100		75
$\underline{a}_{t,d}$ in m/s^2	$-\mu_h g$	0	$-\mu_h g$
$\overline{a}_{t,d}$ in m/s^2	4	0	4
$\overline{a}_{n,d}$ in m/s^2	0	$0.85\mu_h g$	$0.85\mu_h g$
$\underline{a}_{n,d}$ in m/s^2	0	$-0.85\mu_h g$	$-0.85\mu_h g$

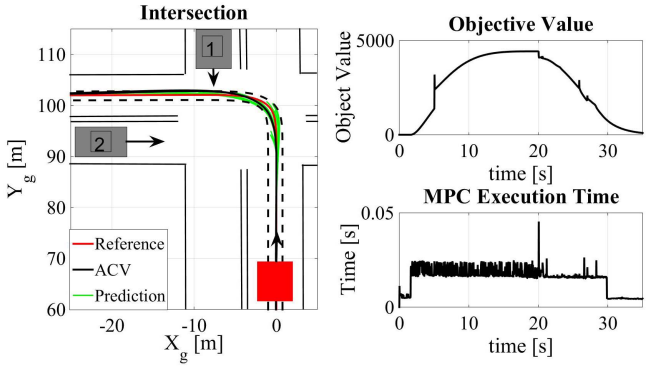

Fig. 9. Intersection: overview, objective value, and MPC execution time.

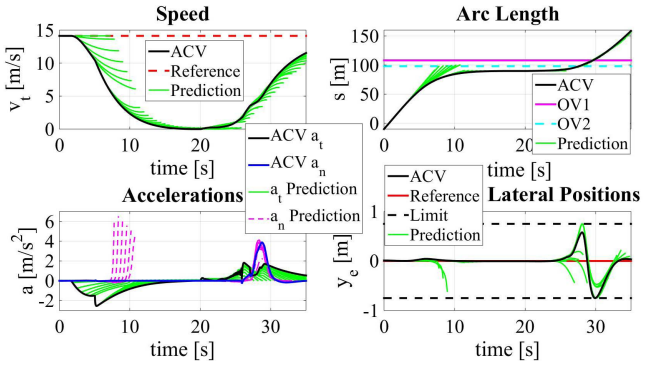

Fig. 10. Intersection: speed, positions, and accelerations.

A. Simulation Results—Fully Autonomous Mode

First, we consider a scenario where the ACV needs to follow a reference path through an intersection with a left turn in the presence of two other vehicles (object vehicles), see Fig. 9. The ACV is restricted with ± 0.75 m tolerance along its reference path. The traffic light in front of it turns red at $t = 5$ s when $s = 90$ m and it turns green at $t = 20$ s. Suppose object vehicle 2 (OV2) violets the red light it faces at $t = 20$ s and keeps going through the intersection from the left to the right. Suppose also that object vehicle 1 (OV1, driven by an attentive driver) yields to OV2 and starts moving forward at $t = 27$ s. The predicted and updated plans for the ACV are shown in Fig. 10. Before $t = 5$ s, PTG plans to slow down the ACV and make a left turn and go through the intersection, therefore, the lateral acceleration a_n magnitude increases in the planning/prediction. After detecting the red light, the plan changes to a further deceleration, and finally, a complete stop in front of the traffic light, constrained by $\overline{s} = 90$ m. When the light turns green, when the arc length restriction is softened to allow the ACV to accelerate. However, as OV2 does not obey its traffic light, PTG plans to yield to it to avoid collision and wait until OV2 has crossed the intersection. Then, the PTG guides the ACV to reduce its acceleration to let OV1 pass the intersection as shown in Fig. 10.

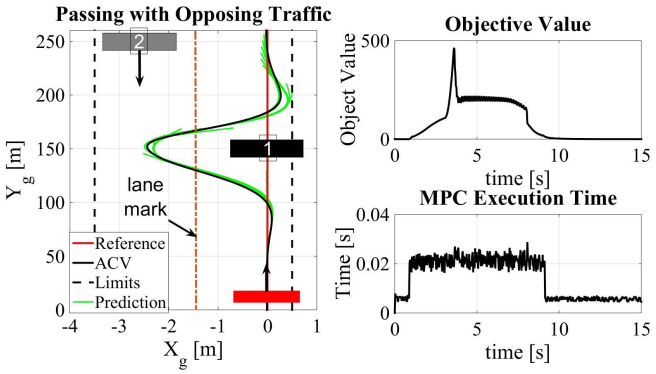

Fig. 11. Passing with opposing traffic: overview, objective value, and MPC execution time.

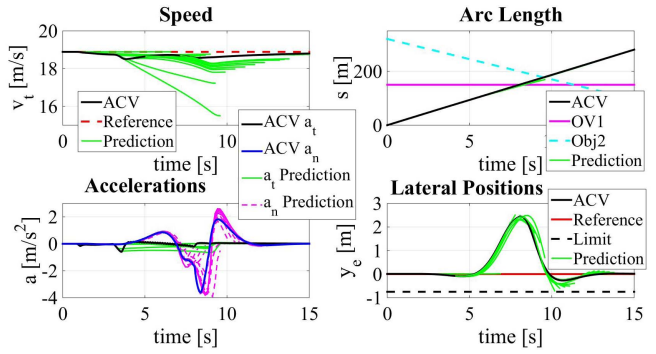

Fig. 12. Passing with opposing traffic: speed, positions, and accelerations.

The second set of results are generated for the ACV in a CA scenario involving two objects shown in Fig. 11: the static OV1 partially occupies the same lane in front of the ACV. Another vehicle OV2 travels in the opposite direction on a second lane. The ACV needs to pass the static OV1 to maintain the reference speed, while avoiding collision with OV2. First, the PTG plans to reduce the speed and wait until OV2 on the left lane has passed. However, when it approaches OV1 and obtain more information about the environment, the PTG finds it possible to pass OV1 before hitting OV2. Thus, it guides the ACV to accelerate, steer to the left to pass OV1 and, then, steer back to the reference lane to avoid OV2. For different initial conditions, the PTG could also force the ACV to wait for OV2 to pass (for such a senario, see [20]).

B. Simulation Results—Semiautonomous Modes

To minimize acronyms in figures 13 and 14, we shall keep using the notation ACV here, even though the operations are semiautonomous. Fig. 13 shows results generated for driving on a straight road in ACC mode with later control maintained by the human driver. In this case, the PTG plans only the vehicle's longitudinal acceleration a_t to track the reference speed. When the speed of the front OV1 is lower than the ACV, the ACV slows down and follows the front object at a distance defined by (25).

Fig. 14 shows results for the same maneuver conditions but with the PTG in LKA and CA semiautonomous modes. In these scenarios, the speed is held constant by the human driver as the ACV approaches the slow moving vehicle in

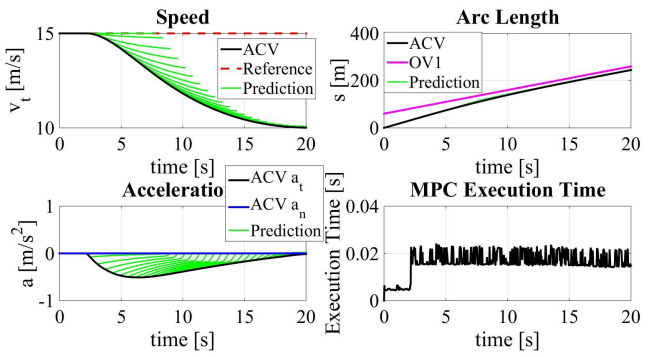

Fig. 13. ACC mode: speed, positions, accelerations, and MPC execution time.

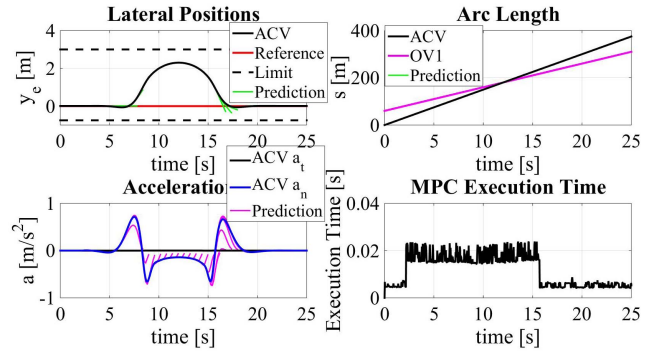

Fig. 14. LKA&CA mode: lateral deviation, positions, accelerations, and MPC execution time.

TABLE II

EXECUTION TIME IN MILLISECONDS ON INTEL i5 4200U NOTEBOOK CPU, 2.5 GHz, ESTIMATED BY THE ACADO SOLVER, $N_p = 40$

Scene	Fully Autonomous		Semi-Autonomous	
	Intersection	Passing with opposing traffic	ACC	LKA&CA
mean	14.8	14.4	16.1	11.9
min	4.3	4.4	4.3	4.1
max	46.4	29.9	24.2	23.9

front (see arc length). The PTG directs the ACV with lateral input $\Delta\dot{\psi}_p$ to pass OV1 in a safe distance of about 2 m until the ACV again tracks the reference with zero deviation. The PTG makes the ACV pass on the left side of the object as the traffic lane constraint prevents it from doing so on the right side.

C. Execution Time of MPC

Table II shows the mean, minimum, and maximum execution times estimated by the ACADO Toolbox for the previously discussed maneuvers. It can be seen that the mean execution times of different operation modes stay within the range of 11–16 ms. A closer examination of the execution time records indicates that the execution time could jump to a higher level when some inequality constraints become active. This is due to the active set method used in the solver where the Hessian matrix of the Lagrangian function needs to be reevaluated once the active constraints are updated. The jumps are usually found when the speed starts decreasing because of obstacle/objects engaging in the inequality constraints.

Especially for the intersection case, a peak execution time of about 46.4 ms is observed when the traffic light turns green and the constraint to avoid the red-light violating OV2 become activated. The largest execution time can be used to guide the selection of the update rate T_{mpc} of the MPC for real-time applications. While further investigation is required for a conclusive remark, these results indicate that the proposed framework for PTG is quite feasible for implementation on modern real-time hardware.

VII. Conclusion

This paper presented a versatile PTG framework for fully autonomous and semiautonomous vehicles feasible for real-time implementation. A nonlinear model of particle motion descriptions and road references expressed in curvilinear Frenet frames is used in the PTG level to formulate and solve a nonlinear MPC problem. The formulation integrates several references, obstacle descriptions, and hard constraints imposed by a traffic management or assigner module for lane limits and traffic signal information, as well as vehicle dynamics and actuation constraints. The control inputs computed at the PTG level can be utilized as control references by vehicle dynamics control (VDC) level in tracking mode or merely treated as planned optimal trajectories in planning mode by a suitable speed and path tracking VDC. A number of simulation results were included to illustrate the performance of the PTG with VDC in tracking mode interfaced with a high-fidelity vehicle model for both fully autonomous and semiautonomous modes in public traffic situations. It is shown that the overall scheme shows good performance in various scenarios with dynamic objects and operating modes. It is also illustrated that the use of the particle motion model in the PTG allows reasonable and practically feasible execution times, while handling active inequality constraints prevalent in dynamic public traffic scenarios. Future work will include extensions of the predictive guidance framework to accommodate uncertainties from environmental conditions, sensor imperfections, and other disturbances.

References

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[2] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1077–1091, Dec. 2005.

[3] A. Liniger and J. Lygeros, "A viability approach for fast recursive feasible finite horizon path planning of autonomous RC cars," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, Apr. 2015, pp. 1–10.

[4] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[5] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency manoeuvres for automated highway system: Obstacle avoidance problem," in *Proc. 36th IEEE Conf. Decision Control*, vol. 5. Dec. 1997, pp. 4774–4779.

[6] I. Papadimitriou and M. Tomizuka, "Fast lane changing computations using polynomials," in *Proc. Amer. Control Conf.*, vol. 1. Denver, CO, USA, Jun. 2003, pp. 48–53.

[7] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, "MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles," in *Proc. Amer. Control Conf.*, Minneapolis, MN, USA, Jun. 2006, pp. 5763–5768.

[8] Z. Shiller and J. C. Chen, "Optimal motion planning of autonomous vehicles in three dimensional terrains," in *Proc. IEEE Int. Conf. Robot. Autom.*, Cincinnati, OH, USA, May 1990, pp. 198–203.

[9] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear model predictive control on long and on short horizons," in *Proc. 52nd Conf. Decision Control (CDC)*, Dec. 2013, pp. 5113–5118.

[10] H. J. Ferreau, T. Kraus, M. Vukov, W. Saeys, and M. Diehl, "High-speed moving horizon estimation based on automatic code generation," in *Proc. 51st IEEE Conf. Decision Control (CDC)*, Maui, HI, USA, Dec. 2012, pp. 687–692.

[11] T. Keviczky, P. Falcone, F. Borrelli, and J. A. Hrovat, "Predictive control approach to autonomous vehicle steering," in *Proc. Amer. Control Conf.*, Minneapolis, MN, USA, Jun. 2006, pp. 4670–4675.

[12] 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.

[13] A. Gray, Y. Gao, J. K. Hedrick, and F. Borrelli, "Robust predictive control for semi-autonomous vehicles with an uncertain driver model," in *Proc. IEEE Intell. Vehicles Symp.*, Gold Coast, Australia, Jun. 2013, pp. 208–213.

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

[15] 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 *Proc. 16th Int. IEEE Annu. Conf. Intell. Transp. Syst.*, Hague, The Netherlands, Oct. 2013, pp. 378–383.

[16] 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. 16th Int. IEEE Annu. Conf. Intell. Transp. Syst.*, Hague, The Netherlands, Oct. 2013, pp. 2335–2340.

[17] H. J. Ferreau, "Model predictive control algorithms for applications with millisecond timescales," Ph.D. dissertation, Arenberg Doctoral School Sci., KU Leuven, Leuven, Belgium, 2011.

[18] G. Prokop, "Modeling human vehicle driving by model predictive online optimization," *Int. J. Vehicle Mech. Mobility, Vehicle Syst. Dyn.*, vol. 35, no. 1, pp. 19–53, 2001.

[19] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical model predictive control framework for autonomous ground vehicles," in *Proc. Amer. Control Conf.*, Seattle, WA, USA, Jun. 2008, pp. 3719–3724.

[20] T. Weiskircher and B. Ayalew, "Predictive trajectory guidance for (semi-)autonomous vehicles in public traffic," in *Proc. Amer. Control Conf.*, Chicago, IL, USA, Jul. 2015, pp. 3328–3333.

[21] T. Weiskircher and B. Ayalew, "Frameworks for interfacing trajectory tracking with predictive trajectory guidance for autonomous road vehicles," in *Proc. Amer. Control Conf.*, Chicago, IL, USA, Jul. 2015, pp. 477–482.

[22] T. Weiskircher and B. Ayalew, "Predictive ADAS: A predictive trajectory guidance scheme for advanced driver assistance in public traffic," in *Proc. Eur. Control Conf.*, Linz, Austria, Jul. 2015, pp. 3402–3407.

[23] Q. Wang, T. Weiskircher, and B. Ayalew, "Hierarchical hybrid predictive control of an autonomous road vehicle," in *Proc. Dyn. Syst. Control Conf.*, Columbus, OH, USA, 2015, p. V003T50A006.

[24] Q. Wang, B. Ayalew, and T. Weiskircher, "Optimal assigner decisions in a hybrid predictive control of an autonomous vehicle in public traffic," in *Proc. Amer. Control Conf.*, Boston, MA, USA, Jul. 2016, pp. 3468–3473.

[25] G. Genta, *Motor Vehicle Dynamics: Modeling and Simulation* (Series on Advances in Mathematics for Applied Sciences), vol. 43. Singapore: World Scientific, 1997.

[26] B. Heissing and M. Ersoy, Eds., *Chassis Handbook*. Wiesbaden, Germany: Teubner, 2010.

[27] M. Abe and W. Manning, *Vehicle Handling Dynamics: Theory Application*. Amsterdam, The Netherlands: Butterworth-Heinemann, 2009.

[28] A. V. Zanten, R. Erhardt, and G. Pfaff, "VDC, the vehicle dynamics control system of bosch," SAE Tech. Paper 950759, 1995.

[29] A. V. Zanten, "Evolution of electronic control systems for improving the vehicle dynamic behavior," in *Proc. Int. Symp. Adv. Vehicle Control (AVEC)*, Hiroshima, Japan, 2002, pp. 1–9.

[30] V. Jain and T. Weiskircher, "Prediction–based hierarchical control framework for autonomous vehicles," *Int. J. Vehicle Auto. Syst.*, vol. 12, no. 4, pp. 307–333, 2014.

[31] J. L. Garriga and M. Soroush, "Model predictive control tuning methods: A review," *Ind. Eng. Chem. Res.*, vol. 8, no. 49, pp. 3505–3515, 2010.

[32] A. S. Yamashita, P. M. Alexandre, and A. C. Zanin, "Reference trajectory tuning of model predictive control," *Control Eng. Pract.*, no. 50, pp. 1–11, May 2016.
[33] P. Stolze, M. Kramkowski, T. Mouton, M. Tomlinson, and R. Kennel, "Increasing the performance of finite-set model predictive control by oversampling," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Cape Town, South Africa, Feb. 2013, pp. 551–556.
[34] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear mpc: A tutorial using acado integrators," *Optim. Control Appl. Methods*, vol. 36, pp. 685–704, 2015.
[35] H. F. Grip, L. Imsland, T. A. Johansen, J. C. Kalkkuhl, and A. Suissa, "Vehicle sideslip estimation," *IEEE Control Syst.*, vol. 29, no. 5, pp. 36–52, Oct. 2009.
[36] D. Heß, M. Althoff, and T. Sattel, "Comparison of trajectory tracking controllers for emergency situations," in *Proc. IEEE Intell. Veh. Symp.*, Gold Coast, Australia, Jun. 2013, pp. 163–170.

Thomas Weiskircher received the Diploma degree in mechatronics engineering from the Universität des Saarlandes, Saarbrücken, Germany, in 2009, and the Ph.D. degree in mechanical engineering from Technische Universität Kaiserslautern, Kaiserslautern, Germany, in 2013.

Since 2014, he has been a Post-Doctoral Researcher in vehicle control with the Applied Dynamics and Control Research Group, Clemson University International Center of Automotive Research, Greenville, SA, USA. His current research interests include optimal control, real-time model predictive control for automotive applications, vehicle dynamics control, and active safety systems.

Qian Wang received the Diploma and master's degrees in vehicle engineering from the South China University of Technology, Guangzhou, China, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree with the Applied Dynamics and Control Research Group, Clemson University, Clemson, SC, USA.

His current research interests include optimal control, predictive motion planning for autonomous vehicles, and control allocation for vehicle actuation system.

Beshah Ayalew received the M.S. and Ph.D. degrees in mechanical engineering from Pennsylvania State University, State College, PA, USA, in 2005.

He is currently a Professor of Automotive Engineering, and the Director of the DOE GATE Center of Excellence in Sustainable Vehicle Systems, Clemson University International Center for Automotive Research, Greenville, SA, USA. His current research interests include the modeling and control of dynamic systems with applications in vehicle systems dynamics, manufacturing processes, and vehicle energy systems. His recent research is funded by various federal and industry grants and contracts.

Dr. Ayalew is an active member of the ASMEs Vehicle Design Committee, the IEEE Control Systems Society, and Society of Automotive Engineers (SAE). He received the Ralph R. Teetor Award from SAE in 2014, the Clemson University Board of Trustees Award for Faculty Excellence in 2012, the NSF CAREER Award in 2011, and the Penn State Alumni Association Dissertation Award in 2005.