2015 American Control Conference
Palmer House Hilton
July 1-3, 2015. Chicago, IL, USA

Frameworks for Interfacing Trajectory Tracking with Predictive Trajectory Guidance for Autonomous Road Vehicles*

Thomas Weiskircher[1] and Beshah Ayalew[1]

Abstract— **This paper investigates two frameworks for interfacing trajectory tracking functions with a computationally tractable nonlinear model predictive trajectory guidance module for an autonomous road vehicle. In the first framework, the predictive trajectory guidance itself is configured in a tracking mode where the control inputs computed by the model predictive control act as targets for some lower-level control system acting on the steering, brakes or engine. In the second framework, the state trajectories computed by the predictive trajectory guidance module are configured to be tracked by a non-predictive state tracking controller derived via input/output linearization. In both frameworks, the main considerations are driven by the time scales selected for computational expediency of the predictive guidance module, its update rate, and for control of the lower-level dynamics. The performance of the two frameworks, including the computational aspects, are compared considering public road driving as well as high performance race line scenarios.**

I. INTRODUCTION

Recent developments in vehicle control show promising results towards guaranteeing safe vehicle operation under various conditions. These are not only of high importance for todays vehicles with the driver-in-the-loop but for future autonomous vehicles. The control formulations for autonomous vehicles often have a natural hierarchy where trajectory guidance is performed as a higher-level control and trajectory tracking tasks are executed as a lower-level control.

In this paper, we focus on how an MPC-based predictive trajectory guidance (PTG) module may be properly integrated with the lower-level vehicle control. The predictive trajectory guidance we use here is proposed in a companion work of the authors [1], but other trajectory guidance algorithms could also be considered [2], [3]. In [1], the MPC scheme of the PTG computes the control inputs needed for the vehicle to follow the optimal state trajectory (position, speed, heading, yaw rate, etc). Then, a lower-level controller generates the requested MPC control input with the available actuation (steering, engine, brakes). In this structure, high MPC update rates may be necessary to handle fast vehicle dynamics. This increases the computational burden in spite of the efficiency of the optimization algorithms that may be selected. Alternatively, the PTG can be configured to output an optimal state trajectory and then various methods can be used at the lower-level control to track the state trajectory. In fact, path and speed tracking control is widely studied in the context of traditional vehicle stability control, where the desired vehicle state trajectories to be tracked are generated by passing human driver inputs through some reference models. However, when it comes to autonomous vehicle control, the integration of the higher-level predictive trajectory guidance and the lower-level state trajectory tracking has not been explored much. Existing works that treat this issue include [3]–[7] where the lower-level tracking controller is an additional MPC formulation employing vehicle dynamics models. Therein, various issues such as uncertain driver models, linearizing approaches, and different vehicle dynamics models are proposed to improve the robustness and to reduce the computational burden of the control framework. However, the challenge with such additional MPC-based tracking controllers at the lower-level is that they require high update rates to accommodate the fast vehicle dynamics being controlled, often using high-fidelity models. This means extra computation resources are necessary for real-time implementation of such solutions.

In light of the above discussion, in this paper, we introduce and analyze two ideas for interfacing the predictive trajectory guidance module to lower-level controllers. We formulate the lower-level control structures as using either the computed control inputs of the PTG or the computed state trajectory as their reference. In the first case, the MPC in the PTG itself is said to be configured in a trajectory tracking mode (despite the name), while in the second case, it only executes position and speed trajectory planning tasks in what we call a planning mode. We discuss the specific modifications needed in the PTG for properly interfacing with the suitable lower-level controllers in each mode. We do not dwell on the design of the lower-level controllers. Rather we adopt existing approaches and analyze the influence on the overall performance of the control framework. In both modes, the main consideration is the time scale separation between the large step sizes needed for computational expediency, and the faster control update rate for the PTG so as to use the latest information in dynamic environments, and the even faster sample times needed for control of the lower-level dynamics.

The rest of the paper is organized as follows: In Section II, the control frameworks with the two different modes of the PTG are introduced and the underlying model for trajectory planning is briefly reviewed. Then, the interfacing considerations are outlined in Section III. Subsequently, Section IV illustrates the performance of the two frameworks using simulations of a high-fidelity vehicle model. Conclusions are offered in Section V.

*This work was supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD)

[1]Thomas Weiskircher and Beshah Ayalew are with the Applied Dynamics & Control Research Group at the Clemson University - International Center for Automotive Research, 4 Research Drive, 29607, Greenville, SC, USA, {tweiski, beshah}@clemson.edu

II. Overall Control Framework and PTG Modes

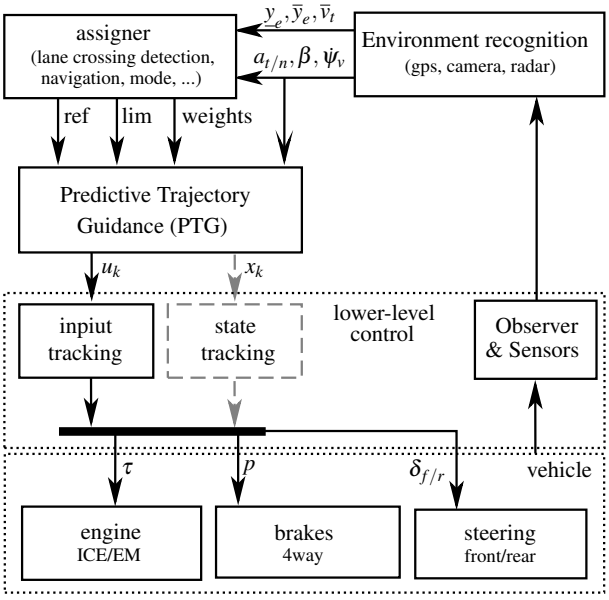

Fig. 1. Multi-level control structure for autonomous driving

Figure 1 shows the overall control framework including the two aforementioned modes of the PTG. For the overall control framework in either mode, the required information from the environmental recognition and vehicle dynamics sensors are assumed to be processed via an assigner module that will deliver all required data about road lane boundaries and the (static and dynamic) objects to avoid. Also, the vehicle position and its alignment to the road are assumed to be available from measurement or estimation/observers.

The PTG is in Tracking Mode (TM) when the PTG itself is configured as a tracking controller that computes the control actions in terms of the desired accelerations and/or yaw rate needed to optimize the MPC objective in the PTG, i.e, to minimize the initial error of the vehicle states and to track a reference path in the future while avoiding obstacles and observing traffic rules. In this case, the corresponding lower-level control acts on u_k (input tracking, left side in Fig. 1). This mode of the PTG was introduced by the authors in [1] with additional considerations for accommodating semi-autonomous vehicle control.

The PTG is in Planning Mode (PM) when the PTG computes a future optimal state trajectory which is passed down to a classical lower-level path and speed tracking controller acting on re-interpretations of x_k (state tracking, right side in Fig. 1). In the PM of the PTG, it desired to decouple the planning from the vehicle dynamics by not re-initializing the prediction to the current state x_0 of the actual vehicle. This avoids reliance on detailed vehicle state information (e.g. side slip) and poses the PTG as an open-loop planner that can be called at reduced sampling rates merely for the purpose of planning the optimal trajectory in the dynamic traffic environment. Detailed depictions of the vehicle trajectories in PM and TM are given in Fig. 5 along with discussions of the lower-level controllers for each mode in Section III.

A. Particle Motion Model for PTG

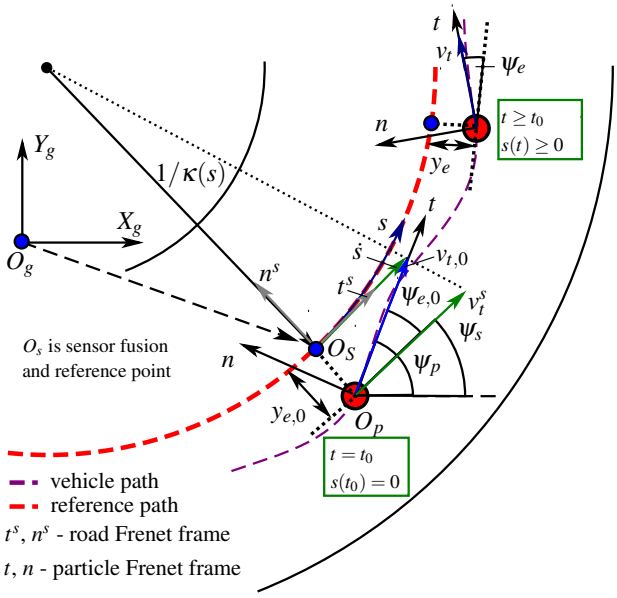

Fig. 2. Definitions of the curvilinear motion of the vehicle

A reduced vehicle model is needed for real-time implementation of the nonlinear MPC of the PTG. For this work, we adopt the particle motion model expressed in the curvilinear frame with the definitions depicted in Fig. 2. The complete model equations are [1]:

$$\dot{v}_t = a_t, \tag{1a}$$

$$\dot{\psi}_e = \dot{\psi}_p - v_t \cos(\psi_e)\left(\frac{\kappa(s)}{1 - y_e \kappa(s)}\right), \tag{1b}$$

$$\dot{y}_e = v_t \sin(\psi_e), \tag{1c}$$

$$\dot{a}_t = 1/T_{a_t}\left(a_{t,d} - a_t\right), \tag{1d}$$

$$\ddot{\psi}_p = 1/T_{\dot{\psi}_p}\left(\dot{\psi}_{p,d} - \dot{\psi}_p\right), \tag{1e}$$

$$\dot{s} = v_t \cos(\psi_e)\left(\frac{1}{1 - y_e \kappa(s)}\right). \tag{1f}$$

Here, the desired acceleration $a_{t,d}$ and the desired yaw rate $\dot{\psi}_{p,d}$ act as the control inputs used to maneuver the particle/vehicle along a given reference path. The curvature of the reference path $\kappa(s)$ is assumed to be known along the reference path coordinate s. v_t, a_t are the longitudinal particle speed and acceleration. $\dot{\psi}_p$ is the yaw rate, ψ_e is the aligning error to the reference path, and $T_{a_t}, T_{\dot{\psi}_p}$ are the time constants of the first order approximations of the longitudinal and lateral controlled vehicle dynamics. $\dot{s}$ is the projected path speed according to Fig. 2. With the knowledge of the reference path curvature $\kappa(s)$ and the assumption that the vehicle is required to follow this path, the following input transformation can be made:

$$\dot{\psi}_{p,d} = \dot{\psi}_{p,r} + \Delta\dot{\psi}_{p,d} = v_t \kappa(s) + \Delta\dot{\psi}_{p,d}. \tag{2}$$

This variable $\Delta\dot{\psi}_{p,d}$, which is a yaw rate correction, and the longitudinal acceleration are the main inputs in the above particle motion model adopted for the vehicle.

B. Formulation of MPC for PTG

For tracking a reference path with the PTG, the objective function for the MPC can be expressed in terms of the tracking errors and control inputs as follows:

$$J = \sum_{k=0}^{N_p} \underbrace{||y_k - r_k||_Q^2}_{\text{tracking error}} + \sum_{k=0}^{N_p-1} \underbrace{||u_k - u_r||_R^2}_{\text{control minimization}} \tag{3}$$

Here, k is the prediction step $k \in (0,1,2,\cdots,N_p)$ with the prediction length $N_p = H_p/\Delta T$, for sample time ΔT and prediction horizon H_p. Q and R include a weighting for each state/input. The MPC algorithm for the PTG solves a nonlinear program at each update interval T_{mpc} ,which is generally selected to be shorter than the ΔT of the MPC model to allow use of the latest traffic information while reducing the problem size for the optimization. The reader is referred to [1] for the details of the constraint formulations and weight selections for the MPC scheme.

In the configuration for the PTG for tracking mode, the initial state x_0 at each MPC update is assumed to be obtained from sensing or estimation. In contrast to this, for the PTG for planning mode, the initial state x_0^{PM} at the MPC update needs special handling. This is because, as mentioned in Section II, it is desired to decouple the planning from the current vehicle dynamics. In particular, we consider the practical case where the ΔT of the MPC model is larger than the MPC update interval T_{mpc}. Then, updated initial conditions are required at the time $t_0 + T_{mpc}$ as the available MPC internal states at $t_0 + \Delta T$ would be too far in the future. A simple linear interpolation of the MPC internal states at the time $t_0 + T_{mpc}$ could approximate the initial conditions, but because of the nonlinear nature of the MPC model, this is likely to lead to an approximation error. Consequently, a copy of the prediction model (1) is used to resolve this issue. The states of the particle motion model are calculated by numerical integration using a fixed step 4th order Runge-Kutta method from t_0 until $t_0 + T_{mpc}$. The integration of the particle motion model starts at the current position s of the real vehicle, and the *control inputs* obtained from the latest MPC execution in the PTG are applied and held constant until the next update:

$$x_0^i = \begin{bmatrix} v_t^i & y_e^i & \psi_e^i & s & a_{t,d}^i & \dot{\psi}_p^i \end{bmatrix}^T, \tag{4a}$$

$$x_0^{PM} = x_0^i + \int_{t_0}^{t_0+t_{mpc}} \dot{x}^i dt \quad \text{with} \quad \dot{x}^i = f(x_0^i, u_0). \tag{4b}$$

Herein, the upper right index marks the states of the particle motion starting from the last MPC update. At the next re-sampling of the MPC, x_0^{PM} is taken as the initial condition.

III. Lower-Level Controllers

In this section, for each proposed operating mode of the PTG, a separate lower-level control setup is presented.

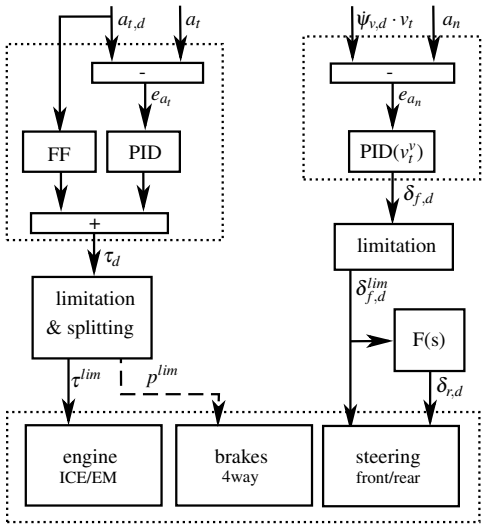

Fig. 3. Lower-level control for trajectory guidance in tracking mode

A. Lower-Level Control for the PTG in Tracking Mode

In this mode, the PTG computes the accelerations and/or yaw rate inputs needed to control the vehicle motion and to track the reference path subject to traffic regulations and obstacle avoidance. Figure 3 depicts the typical elements of a suitable lower-level controller that acts on these accelerations and/or yaw rate generated by the PTG. The left side shows the longitudinal control via feed-forward (FF) and PID-feedback control, while the right side contributes to the lateral vehicle motion control with a speed dependent PID feedback controller. The tuning can be done by following traditional steps. The output of the longitudinal control is the wheel torque, which is subjected to physical limits and assigned for the engine or brakes depending on the sign (traction or braking torque). The FF module uses the vehicle mass m to generate the required torque with $\tau_{FF} = m a_{t,d} r_w$. Herein, the other resistances are neglected to keep the number of required parameters low. The balance may be compensated for by the feedback part.

The torque limitation module first calculates the limits for the longitudinal tire force F_x according to the friction circle:

$$\overline{F}_x = \sqrt{(\mu_H F_z)^2 - F_y^2} \tag{5}$$

$$\tau_d^{lim} = \text{sign}(\tau_d) \cdot \min(\tau_d, \overline{F}_x r_w) \tag{6}$$

with the friction coefficient μ_H, the wheel load F_z, the lateral wheel force F_y, and the wheel radius r_w. For a rear wheel driven vehicle considered as an example, the torque is mapped first to the corresponding axle and then saturated according to this. In case of braking, a fixed split is assumed between the front and rear axles. The output of the lateral acceleration control is a reference steering angle $\delta_{f,d}$ of the front axle steering actuator. While no significant speed dependency was found for the longitudinal control, this is not the case for the lateral acceleration. The steering angle is limited when a given side slip angle of the front tires is reached. Also, a rear steering actuator is included for added

stability function. The following transfer function $F(s)$ is adopted from [8]:

$$\frac{\delta_r}{\delta_f} = F(s) = P_h \frac{1+T_D s}{1+T_1 s}, \tag{7}$$

Herein, P_h, T_D, and T_1 are speed dependent gains based on the vehicle's bicycle model parameters. In this work, the rear steering angle is limited to in-phase (to front) steering only.

B. Lower-Level Control for PTG in Planning Mode

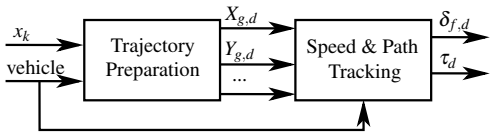

Fig. 4. Lower-level control structure for the PTG in planing mode

The key desirable feature of adopting the planning mode is to take advantage of the information about the collision-free reference vehicle state trajectory computed by the PTG in a dynamic traffic environment. The actual control of the vehicle dynamics to track this planned state trajectory is relegated to the lower-level control depicted in Fig. 4. Herein, a trajectory preparation module interprets the MPC computed state trajectory along with the vehicle position information and outputs the reference state trajectories for the speed and path tracking (SPT) controller block.

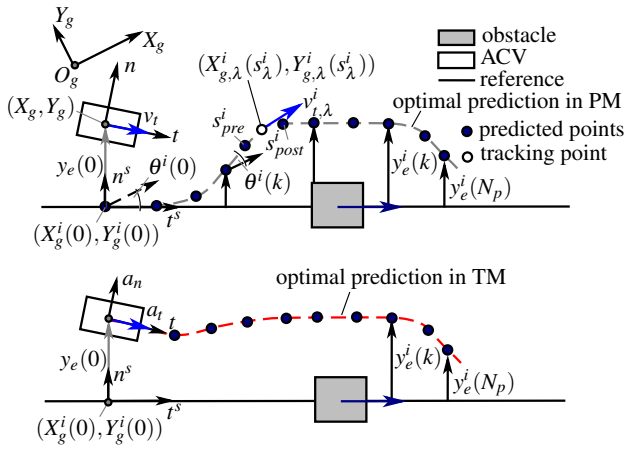

Fig. 5. Resulting trajectories in PM (top) and TM (bottom)

The *Trajectory Preparation* module captures and manipulates the MPC computed states for the complete prediction horizon. In the first preparation step, the arc length of the front decoupling point of the particle is calculated by:

$$s^i_\lambda = s^i + \lambda + v^i_t t + \Delta\lambda. \tag{8}$$

If the initial MPC path coordinate $s(0)$ is reset to zero at each update, s^i in this equation will be removed. As mentioned previously, the MPC update rate of the PTG is in general not as high as that for the lower-level tracking control module. To reconstruct the states in the intermediate fast sampling instances of the lower-level controller, an interpolation part $(v^i_t t)$ is added with t reset to zero after each MPC update. A constant linear speed is considered for the particle model between discrete predicted update instants assuming constant control input of the MPC. Otherwise, the reference position of the particle will be constant until the next MPC update arrives, which leads to a saw tooth effect as the physical vehicle moves forward. Also, the tracking error $\Delta\lambda$ of the last step is stored and added in (8) to prevent steps in the longitudinal control error. The corresponding values for the particle speed, arc length, accelerations, absolute positions and heading direction and rate are needed. Therefore, it is necessary to calculate the absolute path in the global frame O_g using the predicted discrete states as depicted in Fig. 5 (top). The discrete arc length of the ideal vehicle, the corresponding curvature, and the initial heading of the reference path (marked with H_p) are calculated with:

$$v^i_{t,H_p} = \tilde{x}_k(1) \quad \text{for} \quad k = 0..N_p, \tag{9a}$$
$$s^i_{H_p} = \tilde{x}_k(4) \quad \text{for} \quad k = 0..N_p, \tag{9b}$$
$$\Delta s^i_{H_p} = v^i_{t,H_p} \Delta T, \tag{9c}$$
$$\kappa^i_{H_p} = \kappa(s^i_{H_p}), \tag{9d}$$
$$\theta^i(0) = \psi_v - (\psi_e + \beta). \tag{9e}$$

It is obvious that the dimensions of these vectors are related to the number of prediction points N_p of the MPC algorithm. Here, the initial heading of the reference trajectory θ^i_1 is calculated with the vehicle heading, heading error and side slip angle β, respectively. Consequently, the heading angle of the following prediction points are calculated using the reference trajectory of the road lane at the predicted arc length of the particle and the arc length itself. Also, the reference for the real vehicle tracking controller is the MPC predicted path and speed. Thus, the absolute heading of the trajectory is given for $k = 1...N_p$:

$$\theta^i_{H_p}(k) = \theta^i(k-1) + s^i_{H_p}(k)\kappa^i_{H_p}(k) + \psi^i_e(k). \tag{10}$$

Furthermore, the absolute positions result from the actual vehicle position (X_g, Y_g) and

$$X^i_g(0) = X_g + \big(y_e(0) - y^i_e(0)\big)\sin(\theta^i(0)), \tag{11a}$$
$$Y^i_g(0) = Y_g + \big(y^i_e(0) - y_e(0)\big)\cos(\theta^i(0)), \tag{11b}$$

with the initial particle position $(X^i_g(0), Y^i_g(0))$, the predicted particle path deviation y^i_e, and the initial real vehicle path deviation $y_e(0)$. Finally, the absolute reference path position reads (for $k = 1..N_p$):

$$dx = \Delta s^i_{H_p}\cos(\dot{\theta}^i_{H_p}), \quad dy = \Delta s^i_{H_p}\sin(\dot{\theta}^i_{H_p}) \tag{12a}$$
$$\begin{bmatrix} X^i_g(k) \\ Y^i_g(k) \end{bmatrix} = \begin{bmatrix} X^i_g(k-1) \\ Y^i_g(k-1) \end{bmatrix} + \begin{bmatrix} \cos(\theta^i_{H_p}(k-1)) & -\sin(\theta^i_{H_p}(k-1)) \\ \sin(\theta^i_{H_p}(k-1)) & \cos(\theta^i_{H_p}(k-1)) \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{12b}$$

Herein, $\dot{\theta}^i_{H_p}$ can be seen as the change in heading direction per prediction point and (12) describes the rotation of the

incremental changes from the local frame along the reference path. Finally, the corresponding values at s^i_λ result from an interpolation between the previous arc length point s^i_{pre} and the next higher value of the MPC discretization (labeled *post*) via linear interpolation. For instance, the speed of the particle at the front decoupling point is:

$$v^i_{t,\lambda} = v^i_{t,s^i_{pre}} + (v^i_{t,s^i_{post}} - v^i_{t,s^i_{pre}}) \frac{(s^i_\lambda - s^i_{pre})}{(s^i_{post} - s^i_{pre})}. \qquad (13)$$

The same interpolation can be used to find the required reference point values $X^i_{g,\lambda}, Y^i_{g,\lambda}$, $\dot{v}^i_{t,\lambda}$, $\theta^i_\lambda, \dot{\theta}^i_\lambda$ and $\ddot{\theta}^i_\lambda$.

The second module in Fig. 4 is the *SPT controller block*. The particular SPT controller we adopt here uses the coordinates of the so-called front decoupling point $\lambda = J/(l_r m)$ (see Fig. 6). The controller therein is adopted from [9], and is derived via an input/output linearization of the bicycle vehicle model dynamics with a front decoupling point. The advantage of the front decoupling point (also known as center of percussion [10]) over the rear decoupling point is the independence of the steering control law from the rear tire force. However, in the resulting control law, a nonlinear tire force model is used.

The SPT controller block behaves like conventional cascaded position and speed control-loops and aligns the front decoupling point to the desired point on the reference path.It has an outer loop that calculates the required velocities, while the inner loop uses these velocities for the tire force calculation. The SPT block uses the detailed information computed by *Trajectory Preparation* module, see Fig. 4. Note that this model-based tracking controller of the SPT requires more information about the vehicle and tires than the simple acceleration controllers of the tracking mode.

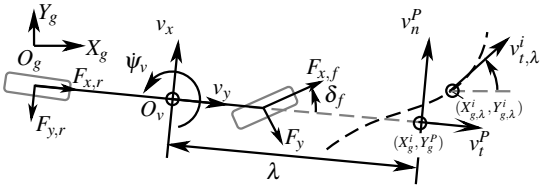

Fig. 6. Trajectory tracking with bicylce model and reference trajectory

IV. Results and Discussions

In this section, the two frameworks involving the PTG modes and the corresponding lower-level controllers are evaluated via simulations. A high-fidelity vehicle model with nonlinear tires, and tire force relaxation, load transfer, driving resistances and actuator dynamics are used for the simulations. The lower-level controllers are sampled with $T_{ll} = 1$ ms commensurate with the fast lower-level vehicle dynamics. For a baseline, we also consider simulations of the particle motion model (labeled *ideal*) given by (1), which uses the direct output u_k of the MPC as its control input.

First, we consider a race line scenario where the PTG needs to plan a reference path through a combination of

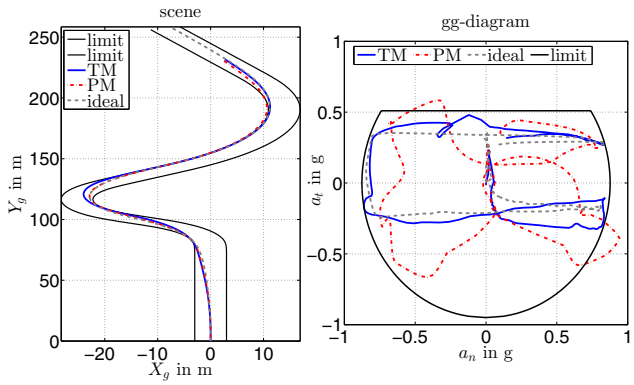

Fig. 7. Race line: absolute path and gg-diagram

corners of a race track. The objective is to find a path that maximizes the driven length in the prediction time H_p. The MPC trajectory guidance pushes the vehicle speed to its maximum under consideration of the vehicle handling and road limitations. The results for the particle motion model and also for the high-fidelity vehicle model with the PTG in tracking mode and in planning mode are depicted in Figs. 7, 8, and 9. The internal discretization of the MPC model is sampled with $\Delta T = 0.15$ s for both modes, while the MPC is sampled with $T_{mpc} = 0.03$ s. This gives in TM a stable control result for all analyzed maneuvers. As can be seen from the gg-diagram, in the TM mode, the lower-level controller makes the high-fidelity vehicle follow the ideal vehicle with minor error. However, the PTG in PM, shows higher deviations from the ideal vehicle's acceleration profile but the position is close to the ideal reference path. While the PTG in TM is able to limit the vehicle lateral position to the given hard constraints (see Fig. 8), the PTG in PM decouples the real vehicle control from the constrained MPC control input and thus its lower-level control violates the limit at $s = 200$ m. The same behavior is observed for the combined acceleration limit which is set to 0.85 g. While the PTG in TM follows the lateral reference very closely (see Fig. 9, upper right plot), the combination of the lateral and longitudinal control with the PTG in PM leads to a limit violation (Fig. 7).

To illustrate the effect of the choice of the update rates in PM, we consider the same race line maneuver with different MPC update rates in the range of $T_{mpc} = 0.03 - 0.125$ s. The results depicted in Fig. 9 (bottom left) show that a higher update rate leads to a slightly better performance provided fast computational hardware are available. An update rate of 0.075 - 0.1 s may be a good compromise in PM.

Next, we consider a cornering maneuver from public driving with reduced lateral dynamics compared to the previous maneuver. In such maneuvers, the control performance of the PTG in PM with its lower-level SPT control may be acceptable. Figure 10 shows the control errors for a simple cornering maneuver with $a_n \leq 4.5$ m/s^2. The lateral error of the center of gravity is used for evaluating the control performance as it is also a state in the PTG module and available in TM. Here, the lateral error of the center of

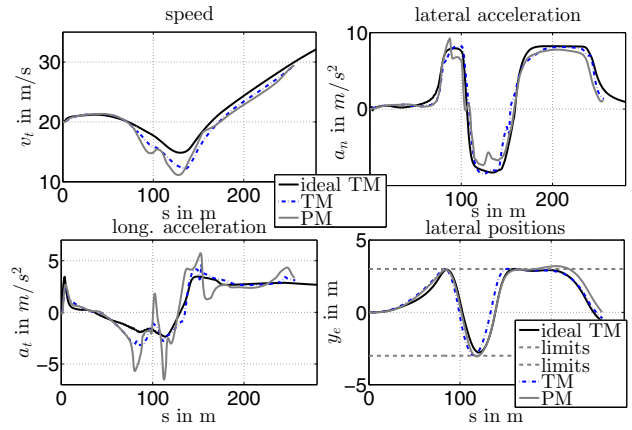

Fig. 8. Race line: speed, accelerations and lateral deviation

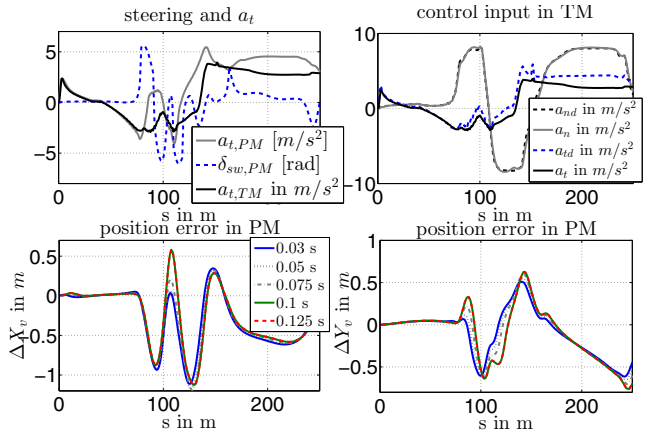

Fig. 9. Race line: control input for PM and TM, lateral and longitudinal control errors for SPT (projected in vehicle coordinate system, different update rates for PTG in PM)

gravity is less than 13 cm for both modes, which seems to be sufficient for this public traffic driving scenario. For the PM, this satisfactory performance also implies slower update rates could be used, while the TM requires the higher sampling of 0.03 s to generate a stable result. For steady state cornering, no clear difference is found in the results of the different sample times. The difference is only seen at the beginning and end of the cornering maneuver where the transients of the lateral vehicle dynamics plays a role.

V. CONCLUSIONS

In this paper, two different ideas for interfacing a Predictive Trajectory Guidance module (PTG) with lower-level vehicle controllers are discussed. The PTG incorporates the particle motion dynamics model, and considers several references and hard constraints imposed by a traffic management module and vehicle dynamics constraints such as tire-road capability. Depending on the lower-level control structure adopted, the PTG can be configured in tracking (TM) or planning mode (PM). The version of the lower-level control for the TM uses the PTG control output to calculate the actuator inputs, while the version in the PM uses the optimal, predicted path and speed of the PTG and

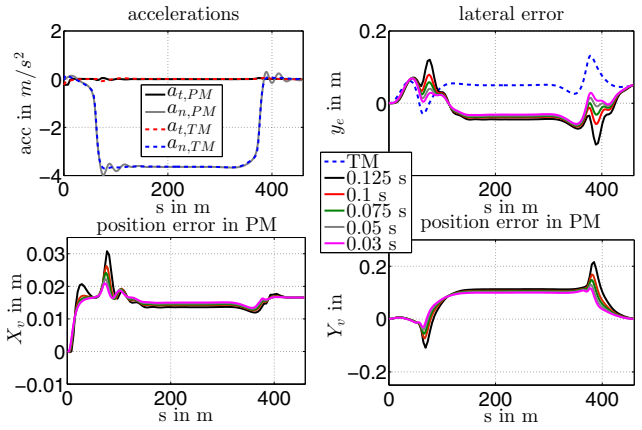

Fig. 10. Cornering in PM/TM: accelerations, control errors of SPT projected in vehicle coordinate system and lateral position error, different update rates for PTG in PM

implements an input/output linearizing tracking controller with a front decoupling point. The control performance of the two configurations have been illustrated for the control of an autonomous vehicle at the limits of handling as well as for simple cornering maneuvers seen on public roads. Using high-fidelity vehicle model simulations, it was observed that while the PTG in TM always operates within the constraints specified, the PTG in PM does not guarantee satisfying path or acceleration limits during aggressive maneuvers. In simple maneuvers, both frameworks could give satisfactory performance with acceptable computational overhead. By contrast, the PTG in PM may avoid the need for oversampling in the MPC computation as the vehicle dynamics is relegated to be stabilized by the lower-level control.

REFERENCES

[1] T. Weiskircher and B. Ayalew, "Predictive trajectory guidance for (semi-)autonomous vehicles in public traffic: Part 1 - framework design," in *American Control Conference*, 2015.

[2] G. Prokop, "Modeling human vehicle driving by model predictive online optimization," *International Journal of Vehicle Mechanics and Mobility : Vehicle System Dynamics*, vol. 35, no. 1, pp. 19–53, 2001.

[3] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical model predictive control framework for autonomous ground vehicles," in *American Control Conference*, 2008, pp. 11–13.

[4] T. Keviczky, P. Falcone, F. Borelli, and J. A. Horvat, "Predictive control approach to autonomous vehicle steering," in *American Control Conference*, Minnesota, USA, 2006, pp. 4670–4675.

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

[6] A. Gray, Y. Gao, J. K. Hedrick, and F. Borelli, "Robust predictive control for semi-autonomous vehicles with an uncertain driver model," in *Proc. of the IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, 2013, pp. 208–213.

[7] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, 2013, pp. 2335–2340.

[8] B. Heissing and M. Ersoy, Eds., *Chassis Handbook*. Teubner, 2010.

[9] D. Hess, M. Althoff, and T. Sattel, "Comparison of trajectory tracking controllers for emergency situations," in *PRoc. of the IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, 2013, pp. 163–170.

[10] K. Kritayakirana, "Autonomous vehicle control at the limits of handling," Ph.D. dissertation, Stanford University, 2012.