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

Control Architecture Design for Autonomous Vehicles

Berntorp, K.; Hoang, T.; Quirynen, R.; Di Cairano, S.

TR2018-125 August 25, 2018

Abstract

The system design of an autonomous vehicle encompasses numerous different interconnected sensing and control algorithms that can be devised in several ways, and the system has to be extensively tested and verified before employed on roads. Full-scale testing of such a system is complex due to the involved time effort, cost aspects, and safety considerations. In this tutorial paper, we give an overview of the design, implementation, and testing of the control stack in autonomous vehicles, based on our research on motion planning and control. We use scaled vehicles as part of the testing and verification of the system design. Scaled vehicles provide possibilities to test some of the relevant interplay in the control stack and robustness to time delays and sensor errors. We illustrate how scaled vehicles can help reduce the amount of full-scale testing, by finding shortcomings of the system design before deploying it on a full-scale test setup.

Conference on Control Technology and Applications (CCTA)

Control Architecture Design for Autonomous Vehicles

Karl Berntorp[1], Tru Hoang[1], Rien Quirynen[1], and Stefano Di Cairano[1]

***Abstract*— The system design of an autonomous vehicle encompasses numerous different interconnected sensing and control algorithms that can be devised in several ways, and the system has to be extensively tested and verified before employed on roads. Full-scale testing of such a system is complex due to the involved time effort, cost aspects, and safety considerations. In this tutorial paper, we give an overview of the design, implementation, and testing of the control stack in autonomous vehicles, based on our research on motion planning and control. We use scaled vehicles as part of the testing and verification of the system design. Scaled vehicles provide possibilities to test some of the relevant interplay in the control stack and robustness to time delays and sensor errors. We illustrate how scaled vehicles can help reduce the amount of full-scale testing, by finding shortcomings of the system design before deploying it on a full-scale test setup.**

I. Introduction

Self-driving cars are decision-making systems whose complexity scales with the level of autonomy. In the highest level of autonomy, Level 5, the self-driving car should be able to provide full-time operation of all aspects of driving under different roadway and environmental conditions. The control architecture and its components can be designed and interconnected in different ways, but, at high level, the system structure of a vehicle with at least partial self-driving capabilities resembles that of Fig. 1 [1]. Each block in Fig. 1 typically consists of several subcomponents, with various communication and sensor interfaces connecting each block [2], [3]. The sensing and mapping module uses various sensor information, such as radar, Lidar, camera, and global positioning system (GPS), together with prior map information, to estimate the parts of the surrounding environment relevant to the driving scenario. The motion-planning block can be abstracted to include a route planner that finds a route on the road network, a discrete decision layer that determines the local driving behavior in terms of, for example, whether to stop at an intersection or to change lane, and a trajectory planner that determines a reference trajectory the vehicle should follow. The vehicle-control block computes adjusted control commands and tracks the reference trajectories from the motion planner. The control commands are subsequently realized by the actuator control that is responsible for the low-level steering and acceleration torque commands. While autonomous vehicles increasingly begin testing on public roads, production vehicles are more commonly being equipped with advanced driver-assistance systems (ADAS) such as adaptive cruise control and lane-change assist. This is driven by both safety and economic aspects such as the high number of traffic accidents associated with overtaking and lane-change maneuvers and potential fuel savings [4].

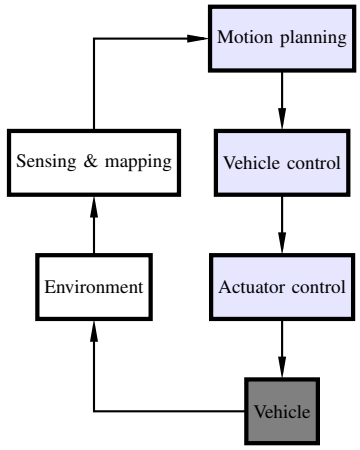

Fig. 1. A high-level view of the system architecture of an autonomous vehicle from a control perspective, with the control and guidance stack in blue color. The different blocks can be interconnected in various ways but the main building blocks remain the same.

There are a number of key issues to solve before self-driving cars become fully operational on public roads. For instance, designing a control stack that is inherently safe, and with redundancy to handle cases when one or several components fail, is a challenge by itself, as illustrated in the DARPA Grand and Urban Challenges [2], [3], [5]–[7]. However, also verifying in practice that the control system is indeed safe further complicates the task. A bottle-neck in designing algorithms for enabling self-driving capabilities is the amount of testing and verification of the different components contained in Fig. 1 [8]. For full-scale tests, the cost of maintaining testing infrastructure and personnel, implementing safety precautions, as well as the time for deploying the algorithms on the vehicle, may have a significant impact on the time for developing the autonomous vehicle.

Computer simulations help in designing the algorithms, but cannot entirely capture unmodeled failure modes and circumstances [9]. Another way to complement full-scale testing is to use scaled vehicles that resemble regular vehicles in terms of kinematics, computation, and sensing capabilities, but without enforcing the strict safety regulations and costs associated with full-scale testing. While scaled vehicles lack the ability to test the self-driving system under every potential environment, they can help in reducing the amount of time needed to be spent on full-scale testing while

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

still capturing the relevant interplay between the different controllers and the connections to other building blocks of an autonomous vehicle. For instance, the interaction between the motion planner and vehicle controller, the driving behavior, and the robustness to timing delays and sensing errors can be evaluated using scaled vehicles.

Scaled vehicles have been used to verify parts of the control stack in previous works, see, for example, [10]–[12]. Our recent research on vehicles with self-driving capabilities has been on estimation [13], [14], motion planning [15]–[17], and vehicle control algorithms [18], [19], which are all important components for self-driving cars. In this paper, we give an example design of the control architecture of an autonomous vehicle. Based on some of our recent research on control and estimation algorithms for enabling self-driving cars [15], [18], [20], we show how a scaled vehicle platform can be used to test and verify the operation of the control stack. Based on the structure in Fig. 1, we describe a systematic way of designing the different control blocks, and estimators are designed to increase robustness of the system to sensing errors and inherent timing delays. Specifically, we leverage a recently developed sampling-based motion planner [15] for generating desired trajectories and model predictive control (MPC) [21], [22] for tracking the desired trajectories. We point out important implementation details for achieving good performance in practice and evaluate different aspects of the control performance.

The rest of the paper starts with a description of our vehicle test setup in Sec. II, and Sec. III discusses the modeling and estimation aspects involved when designing the control stack. Sec. IV provides the design of the planning and control layer. The section focuses on the motion planning and real-time vehicle control aspects of the system design. The paper continues with an experimental evaluation in Sec. V and is concluded in Sec. VI.

II. Scaled Vehicle Experimental Platform

We use the Hamster platform [23] for testing and verifying our control stack before deploying on a full-scale vehicle, see Fig. 2. The Hamster is a 25×20 cm mobile robot for research and prototype development. It is equipped with scaled versions of sensors commonly available on full-scale research vehicles, such as a 6 m range mechanically rotating 360 deg Lidar, an inertial measurement unit, GPS receiver, HD camera, and motor encoders. It uses two Raspberry PI3 computing platforms, each with an ARM Cortex-A53 processor running Linux Ubuntu for processing. The robot has Ackermann steering and is therefore kinematically equivalent to a full-scale vehicle, and its dynamics, such as the suspension system, resembles that of a regular vehicle. Our setup consists of multiple Hamster robots. The Hamster has a low-level controller with dedicated hardware for power distribution and monitoring. By default, the Hamster is controlled by setting the desired wheel-steering angle and longitudinal velocity. In addition, the Hamster has built-in mapping and localization capabilities, and object detection and tracking can be done with the onboard Lidar and/or

Fig. 2. The Ackermann-steered Hamster mobile robot used in the experiments. The markers (five visible in the figure) are used to track the robot via an Optitrack motion-capture system (Fig. 3).

Fig. 3. A camera from the Optitrack motion-capture system used for verifying our control and estimation algorithms.

camera. Hence, the platform is a good proxy for verifying dynamic feasibility and for testing the performance of the system in a realistic setting, with a sensor setup similar to the one expected in full-scale autonomous vehicles. The Hamster communicates and connects to external algorithms using the robot operating system (ROS).

To be able to evaluate the control and estimation algorithms in terms of tracking errors and resulting trajectories in a controlled environment, we use an Optitrack motion-capture system [24]. The Optitrack system is a flexible camera-based (see Fig. 3) six degrees-of-freedom tracking system that can be used for tracking drones, ground, and industrial robots. Depending on the environment and quality of the calibration, the system can track the position of the Hamster within 0.9 mm and with a rotational error of less than 3 deg. The OptiTrack can be connected to the ROS network using the common VRPN protocol.

III. Modeling and Estimation

We refer to the autonomous vehicle as the ego vehicle (EV). Other moving entities in the region of interest (ROI) of the EV are designated as other vehicles (OV). The OVs can be either in autonomous or manual mode. The modeling of the EV can be done either with respect to the global inertial frame, or in local vehicle frame coordinates with respect to a road-aligned noninertial frame. In a practical implementation the alternatives have different limitations depending on the particular driving scenario. In the following, we model the vehicle with respect to the global inertial frame.

The planning and control schemes typically employ receding horizon implementations, in that they rely on models of the environment and the EV itself to predict its evolution in relation to the environment, depending on the control inputs. There is a tradeoff between the complexity and the accuracy of the models, which has to be taken into consideration when choosing the models. In this section we describe a specific

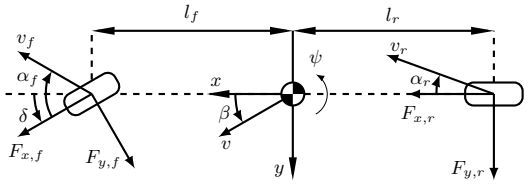

Fig. 4. A schematic of the single-track model and related notation.

choice of EV and OV motion models, but the algorithms are general and hence applicable to any EV and OV model.

A. Vehicle Model

For the vehicle, there are three categories of models; point-mass models that represent the vehicle as a particle; kinematic models that only consider the geometry of the vehicle; and dynamic models that account for the force-mass balances and tire models to more accurately capture the vehicle motion under (highly) dynamic maneuvers. Although a model based on force-mass balances is generally more accurate than a kinematic model, the differences are small for regular driving [25], and model errors are corrected for by the feedback control functions (Fig. 1). Furthermore, a dynamic model depends on more parameters, such as the wheel radii, tire stiffness, and vehicle mass and inertia, which typically are unknown/uncertain and may be difficult, or at least tedious, to estimate. Under the assumption of normal driving conditions (i.e., not at-the-limit maneuvers) the modeling can be based on a single-track model, see Fig. 4, where the two wheels on each wheel axle are lumped together. Hence, in this paper we use the kinematic single-track model

$$\dot{x} = \begin{bmatrix} \dot{p}_X \\ \dot{p}_Y \\ \dot{\psi} \\ \dot{v}_x \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v_x \cos(\psi + \beta)/\cos(\beta) \\ v_x \sin(\psi + \beta)/\cos(\beta) \\ v_x \tan(\delta)/L \\ u_1 \\ u_2 \end{bmatrix}, \tag{1}$$

where p_X, p_Y is the longitudinal and lateral position in the world frame, respectively, $\dot{\psi}$ is the heading rate of the vehicle, v_x is the longitudinal velocity of the vehicle, δ is the steering angle of the front wheel, $L := l_f + l_r$ is the wheel base, and β is $\beta := \arctan\left(l_r \tan(\delta)/L\right)$ is the body-slip angle. The inputs u_1, u_2 are the acceleration and steering rate, respectively. This choice of control inputs allows to provide smooth velocity and steering profiles and to constrain the allowed rate of changes of the velocity and steering angle. In short, we write (1) as

$$\dot{x} = f(x) + g(x)u. \tag{2}$$

We impose various state and input constraints on the vehicle motion. For instance, the steering angle δ is subject to the linear constraint

$$\delta_{\min} \leq \delta \leq \delta_{\max}. \tag{3}$$

This constraint is determined by the physical limitations of the vehicle (i.e., maximum steering angle) or induced by ensuring that the assumptions made for deriving (1) hold. For instance, the linear single-track model is valid for lateral accelerations up to approximately $0.4g$ on dry asphalt, where g is the gravitational acceleration. Hence, limitations on δ can be set by the relation to $\dot{\psi}$ for steady-state cornering [26].

The input constraints on the steering rate $\dot{\delta}$ and acceleration $\dot{v}_x$ are formulated as linear constraints similar to (3),

$$\dot{\delta}_{\min} \leq \dot{\delta} \leq \dot{\delta}_{\max}, \tag{4a}$$

$$\dot{v}_{x,\min} \leq \dot{v}_x \leq \dot{v}_{x,\max}, \tag{4b}$$

where the bounds in (4) are determined as a tradeoff between the allowed level of aggressiveness and driving comfort.

B. Environment Model

For a trajectory-generation system it is necessary to determine the time-varying obstacle space

$$\mathcal{X}_{\text{obs}}(t), \tag{5}$$

which essentially boils down to predicting the motion of the moving obstacles, as well as expressing the time-varying and nonconvex road boundary constraint. Without motion prediction, the motion planner and vehicle control have to assume a static environment, which is unrealistic because the environment is typically dynamic and highly uncertain.

Motion prediction (potentially including driver-intention recognition) can be approached in several ways. For example, deterministic methods predict a single future trajectory, while stochastic methods represent the future trajectories with probability density functions (PDFs), which are estimated using statistical methods such as Monte-Carlo sampling [27]. Another common approach is to base the prediction on Markov chains [25], [28] for reachable set computations [29]. The road boundaries and OVs in the ROI impose constraints on the position of the EV. The road-boundary constraint can be written as

$$\Gamma(p_X, p_Y) \leq 0, \tag{6}$$

where the function Γ is typically constructed from point-wise data of the road and lane boundaries, for instance, obtained from map data or cameras. In general the constraints due to the OVs can take any shape. For instance, if the motion of the OVs is estimated by means of Kalman filters, a natural choice is to model the OVs as ellipsoids. Alternatively, the OVs can be modeled as having rectangular shape [30], or be represented by particles [27], [31].

The constraints resulting from (5), (6) are in general nonconvex, and sometimes it may be too computationally demanding to enforce such constraints in real-time in the vehicle control. In that case, it is possible to enforce (5), (6) with an additional margin in the motion-planning block, which has more computation time available, and then make sure that the vehicle stays within the same margin from the reference trajectory generated by the motion planner [18], thus enforcing the constraints.

In this paper, we model the OVs by assuming that they do not change lane in the planning horizon of the motion planner, and that the velocity change within one planning phase is approximately constant. The predicted trajectories of the OVs are generated by designing lane-keeping controllers for each OV, and uncertainty is incorporated by increasing the size of the box surrounding the vehicle [30]. The predicted trajectories are then used to create the obstacle region (5), which is used to constrain the trajectories generated by the motion planner. This approach is suitable if the update rate of the motion planner is sufficiently fast. Another benefit with such an approach is that the motion planner we employ can straightforwardly find solutions to nonlinear/nonconvex problems. As long as the vehicle control stays within the tolerance levels of the path, safety is therefore guaranteed for the whole system.

C. Sensor-Offset Compensation

Several of the sensors found in the current generation of production vehicles are typically of low cost and as a consequence prone to time-varying offset and scale errors, and may have relatively low signal-to-noise ratio. For instance, the lateral acceleration and heading-rate measurements are known to have drift and large noise in the sensor measurements, leading to measurements that are only reliable for prediction over a very limited time interval. Similarly, the sensor implicitly measuring the steering angle of the wheels has an offset error that, when used for dead reckoning in a vehicle model, leads to prediction errors that accumulate over time.

Knowing the steering angle is crucial for ADAS and autonomous vehicles, because the vehicle will operate without human intervention for long periods of time. Without steering offset compensation, the vehicle control may become unstable, or at least perform poorly, as errors accumulate. Our method for steering offset compensation is based on the vehicle model (1) with added Gaussian process noise to account for modeling errors. We model the steering angle offset b_δ with a random walk model,

$$\dot{b}_\delta = w_\delta, \tag{7}$$

where w_δ is assumed Gaussian distributed process noise. The resulting six-dimensional nonlinear model consisting of (1), (3), and (7) is estimated with an extended Kalman filter, which at each time step k produces a state estimate $\hat{x}_{k|k}$ that is used in the vehicle control to compensate for the sensor offset.

IV. Planning and Control

In this section we briefly discuss the control architecture, and specifically the motion planning and vehicle control aspects. Referring to Fig. 1, the actuator control is handled in the low-level controllers in the scaled vehicle and is not part of this evaluation. Similarly, the route planning, sometimes inside the motion-planning block, is not treated in this paper.

A. Motion Planning

The objective of the motion planner is to determine a collision-free trajectory the vehicle should follow based on the outputs from the sensing and mapping module, while obeying the vehicle dynamics and reaching the goal region $\mathcal{X}_{\text{goal}}$ determined by the decision-making module. The goal region $\mathcal{X}_{\text{goal}}$ can be a specific location on the road or a target set, such as a desired lane.

Sampling-based methods, such as RRT and its variants [32], have been subject to much research over the last two decades, and have been used successfully in autonomous vehicles [33]. Sampling-based methods are focused toward incrementally building a feasible path, or a sequence of feasible paths that converge to an optimal path, given enough computation time. RRT is an important instance of sampling-based methods, which has found various applications [32]. RRT-type algorithms incrementally build a tree by selecting a random sample and expanding the tree towards that sample. Checking if the sample and the corresponding edge is in $\mathcal{X}_{\text{obs}}$ amounts to pointwise comparison. Therefore, RRT does naturally integrate with some of the methods for determining the drivable space, since it does not require a geometric expression for $\mathcal{X}_{\text{obs}}$, as opposed to some graph-search methods, during the construction phase.

In [15], we have developed a variant of the RRT based on particle filtering (PF) for generating complex trajectories, which optionally includes decision making embedded in the planner. Particle filtering is a sampling-based technique for solving the nonlinear filtering problem. The particle filter (PF) numerically approximates the PDF of the variables of interest given the measurement history, by generating N random trajectories and assigning a weight q^i to each one according to how well they predict the observations. The planner relies on the fact that driving requirements, such as staying on the road, right-hand traffic, and avoid obstacles, are known ahead of planning. The driving requirements are modeled as measurements generated by an ideal system. An interpretation is then that the PF determines trajectories and scores them according to how likely they are to obey the driving requirements. In each planning phase, the PF approximates the joint probability density function of the state trajectory conditioned on the decision and driving requirements.

Specifically, our method formulates the vehicle model (1) and driving requirements in discrete time as

$$x_{k+1} = \bar{f}(x_k) + \bar{g}(x_k) w_{x,k}, \tag{8a}$$

$$z_k = h(x_k, m) + w_{z,k}, \tag{8b}$$

where $\bar{f}$ and $\bar{g}$ are discretized versions of (2) and k is the discrete time index. The function h models the driving requirements and is dependent on the chosen decision m by the decision maker. The terms $w_{x,k}$ and $w_{z,k}$ are disturbances on the vehicle model and driving requirements, respectively. They are important to include for several reasons, for example, due to sensor noise from estimation algorithms responsible for estimating the vehicle state and road map

and to avoid infeasibility in trying to fulfill all driving requirements exactly. The control inputs are in our approach contained in the disturbance term $w_{x,k}$.

Using a Bayesian framework, (8a) and (8b) can be reformulated as

$$x_{k+1} \sim p(x_{k+1}|x_k), \tag{9a}$$

$$z_k \sim p(z_k|x_k, m), \tag{9b}$$

where x_{k+1} and z_k are regarded as samples from the respective distributions. Given the vehicle dynamics (1), the goal of the motion planner is to generate an input trajectory u_k, $k \in [0, T_f]$ over the planning horizon T_f satisfying the input constraints (3), (4), such that when applied to (1) leads to a trajectory y_{des}, $k \in [0, T_f]$ that obeys (6), ends up in $\mathcal{X}_{\text{goal}}$, and avoids the obstacle set (5). The PF estimates the PDF $p(x_{0:k}|z_{0:k})$ in each tree expansion toward the goal region. In this paper we generate the desired trajectory by extracting the minimum mean-square estimate,

$$y_{\text{des}} = \sum_{i=1}^{N} q_k^i x_{0:k}^i, \tag{10}$$

in each tree expansion. This procedure is repeated until a solution is found. For more details, see [15]. For our purposes, we have implemented a high-level target-point generator to create the goal region $\mathcal{X}_{\text{goal}}$.

1) Implementation Aspects: Because of sensing errors and unpredicted changes in the environment, for instance, due to new obstacles (5) entering the ROI, we implement the motion planner using a receding-horizon strategy. That is, the computed trajectory is T_f long but is only applied for the time period $\Delta t \leq T_f$, and the allocated computation time for finding the motion plan is δt. We keep a committed tree-branch, where the leaf coincides with the root node of the next planning phase. The part of the tree that does not originate from the end node is deleted. Similar to [18], we propagate tracking-error information from the vehicle control to the motion planner. This helps in determining when to discard the current motion plan and restart the planner. For instance, when the vehicle controller cannot maintain the vehicle closer than some predefined threshold from the desired trajectory y_{des}, the current motion plan is discarded and a new plan is computed, starting with an empty tree.

Another aspect is the node to which we associate the current position within the tree. At the start of the planning phase the motion planner acquires the EV state. The allocated computation time of the motion planner is δt s. Hence, to associate the node in the tree with the position the vehicle will be in when starting to apply the generated motion plan, we predict the current estimated EV state for δt s using the vehicle model (1) and the last control value, and associate the predicted EV state with the closest node in the tree, which becomes the root node of the next planning phase and allows reusing at least part of the previously-computed tree.

B. Vehicle Control by Model Predictive Control

MPC [21], [34], [35] has recently evolved as an important approach in the research literature for automotive control in general and for vehicle-dynamics control in particular. MPC solves at each time step a finite-horizon, possibly nonlinear and nonconvex, optimal-control problem (OCP) and applies the resulting control inputs to the system until the next sampling time step. The MPC formulation depends on the nature of the model, constraints, computational resources, and performance guarantees. Most of the standard nonlinear optimization tools are impractical in such a safety critical embedded application as autonomous vehicle control. The nonlinear and nonconvex OCP needs to be solved at each sampling time instant under stringent timing requirements. For this purpose, tailored continuation-based online algorithms have been developed for solving these nonlinear OCPs in real-time [36].

A common OCP formulation is

$$\min_{x(\cdot),\, u(\cdot)} \quad \int_0^T \|F(x(t), u(t)) - y_{\text{des}}(t)\|_W^2 \, \mathrm{d}t \tag{11a}$$

$$\text{s.t.} \quad 0 = x(0) - \hat{x}_0, \tag{11b}$$

$$0 = f(\dot{x}(t), x(t), u(t)), \qquad \forall t \in [0, T], \tag{11c}$$

$$0 \geq h(x(t), u(t)), \qquad \forall t \in [0, T], \tag{11d}$$

$$0 \geq r(x(T)), \tag{11e}$$

whose objective is tracking the desired trajectory y_{des} from the motion planner. The objective in (11a) consists of a nonlinear least-squares type Lagrange term. For simplicity of notation, T defines both the control and prediction horizon length and we do not consider a terminal (i.e., Mayer) cost term. Note that the NMPC problem depends on the current state estimate $\hat{x}_0$ through the initial condition (11b). The vehicle dynamics in (11c) are described by an implicit system of ordinary differential equations (ODE), which allows the formulation of kinematic, as well as single- and double-track vehicle dynamics as described in [37]. However, in this paper we for simplicity use the vehicle model (2), which is an explicit ODE. Eqs. (11d) and (11e) denote respectively the path and terminal inequality constraints.

The constraints (11d) in the NMPC problem formulation consist of geometric and physical limitations of the system. Depending on the particular maneuver and on which constraints are handled by the motion planner, one can include constraints on the longitudinal and lateral position of the vehicle. However, in practice, it is important to reformulate these requirements as soft constraints because of unknown disturbances and model approximations and errors. For simplicity, we define a quadratic penalization of the slack variable to ensure a feasible solution whenever possible. The steering angle, steering rate, and acceleration constraints (3), (4) are included in (11d).

The cost function in (11a) allows for formulating any standard tracking-type objective. In the presented experimental results, the NMPC scheme is based on a direct tracking of a reference trajectory for the state and control variables

$$\|x(t) - y_{\text{des}}(t)\|_Q^2 + \|u(t) - u_{\text{des}}(t)\|_R^2 + \gamma\, e_y^2(t), \tag{12}$$

where Q and R are the corresponding weighting matrices of suitable dimensions and γ is a scalar weight. The term

$$e_y(\cdot) = \cos(\psi_{\text{ref}})(p_Y - p_{Y,\text{ref}}) - \sin(\psi_{\text{ref}})(p_X - p_{X,\text{ref}}) \tag{13}$$

is the projected distance from the reference trajectory, which ideally should be zero.

1) Online Tracking of the Desired Trajectory: Similar to the work in [18] and based on [26], we model the reference trajectory y_{des} from the motion planner as a piecewise clothoidal trajectory, such that the desired yaw angle ψ_{des} and yaw rate $\dot{\psi}_{\text{des}}$ of the vehicle can be described as

$$\ddot{\psi}_{\text{des}} = v_{\text{des}}(t)\dot{\kappa}(t), \tag{14}$$

where $v_{\text{des}}(t)$ denotes the reference velocity, and $\dot{\kappa}(t)$ denotes the curvature of the reference trajectory.

2) Implementation Aspects: The nonlinear, nonconvex problem (11) renders analytical solutions intractable. Instead, we transform the infinite dimensional OCP (11) into a nonlinear program (NLP) by a control and state parameterization. A popular approach is based on the direct multiple shooting method from [38]. We formulate an equidistant grid over the control horizon consisting of the collection of time points t_i, where $t_{i+1} - t_i = \frac{T}{N} =: T_{\text{s}}$ for $i = 0, \ldots, N-1$. Additionally, we consider a piecewise constant control parametrization $u(\tau) = u_i$ for $\tau \in [t_i, t_{i+1})$. The time discretization for the state variables can then be obtained by simulating the system dynamics using a numerical integration scheme. This corresponds to solving the following initial value problem

$$0 = f(\dot{x}(\tau), x(\tau), u_i), \quad \tau \in [t_i, t_{i+1}], \quad x(t_i) = x_i. \tag{15}$$

We employ a tailored implementation using the open-source `ACADO` code generation tool [39]. The nonlinear optimal control solver in this toolkit uses an online variant of Sequential Quadratic Programming (SQP), known as the Real-Time Iteration (RTI) scheme [40]. Under some reasonable assumptions, the stability of the closed-loop system based on the RTI scheme can be guaranteed also in presence of inaccuracies and external disturbances [40]. `ACADO` Toolkit exports efficient, standalone C-code implementing the RTI scheme for fast optimal control. It supports exploiting specific model structures as detailed in [39]. Specifically, we use the recently proposed `PRESAS` solver [22], which applies block structured factorization techniques with low-rank updates to preconditioning of an iterative solver within a primal active-set algorithm, which results in an efficient solver suitable for embedded automotive applications. For real-time applications, a primal active-set approach has the advantage of providing a feasible, even though suboptimal, solution when being terminated early.

Furthermore, we compensate for the timing delays due to actuators commands and communication through the ROS network by letting the NMPC use the predicted state values instead of the most recent state estimate, using a buffer of the past few control values. The prediction is based on the kinematic vehicle model (1). This time-delay compensation is important for ensuring that the NMPC does not use old information in the feedback control, which otherwise may lead to sluggish performance and even instability.

V. Results

We evaluate the system architecture using our scaled vehicle testing system (Sec. II). We use three Hamster robots in the experimental validation, one acts as the EV and the other two act as OVs. The objective is to avoid the obstacles while circulating a two-lane closed circuit, with the left lane as preferred lane. The track is designed as a super-ellipsoidal track with the size of roughly 3×5 m. In this paper the OVs are controlled by PID controllers that track the designated lane, but in principle also the OVs can use the proposed control architecture. The goal region $\mathcal{X}_{\text{goal}}$ is a circle with radius 0.1 m from the target point. The road boundary constraint (6) is given by an analytic function expressed as super-ellipses, and checking for feasibility of (6) amounts to inequality satisfactions. In the experimental evaluation we use a six minutes long data set. There is one OV in each lane, both with the constant reference velocity $v_{\text{nom}} = 0.2$ m/s. Because one of the robots is in the inner lane, they have different lap times and both regular lane change situations and situations when both lanes are blocked occur. The EV has a reference velocity of $v_{\text{nom}} = 0.4$ m/s, which is sent as a desired velocity to the motion planner.

Fig. 5 shows snapshots of a situation when the two obstacles block both lanes. The sequence of snapshots lasts for about 60 s. Red dots correspond to the planned trajectory for the EV and the green dots are the particles generated in the planning phase. First, a trajectory that overtakes the OV in front of the EV is computed ($t = 22$ s). Then, the EV slows down and stays behind until an opening appears ($t = 66$ s), and the EV moves back to the preferred lane.

The planned (red) and resulting (black) steering and velocity profiles are shown in Fig. 6 for the entire experiment. The measured trajectories closely match the corresponding planned quantities, which shows that the motion planner computes dynamically feasible (i.e., drivable) trajectories. The portion of the data set corresponding to the situation in Fig. 5 is highlighted by the blue dashed lines, and the path of the EV for the entire experiment is shown in Fig. 7. Fig. 8 displays the predicted steering and velocity profiles resulting from the NMPC solution and the corresponding reference profiles from the motion planner, over the prediction horizon of the NMPC. The figure shows the results for every third planning iteration. The NMPC profiles mostly match well with the planner profiles, which indicates that the proposed architecture provides reliable driving behavior. There are a few discrepancies, which are mainly due to model differences. First, the vehicle model in the NMPC uses a first-order actuator model, which is not captured in the motion planner. Second, the steering offset compensator affects the control performance, since it is used as a feedforward term to the NMPC. This causes some deviations in the steering profiles. However, the lower-level NMPC, as also indicated

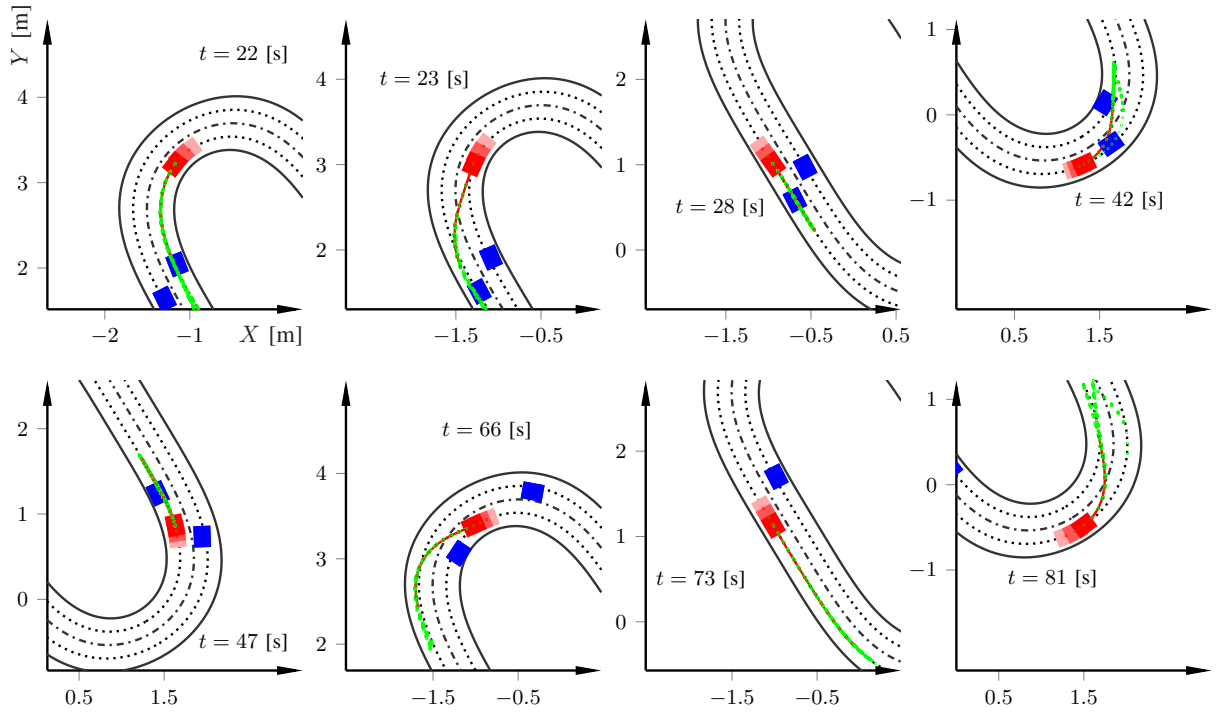

Fig. 5. Eight snapshots from the experimental validation. The EV in red, OVs in blue, particles from the motion planner in green, and corresponding planned trajectory (red dotted). In every figure, snapshots of the EV are shown in increasing color.

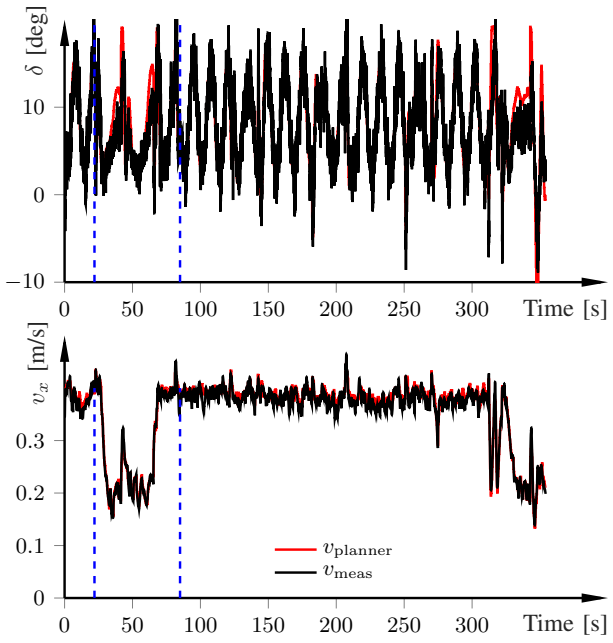

Fig. 6. Steering and velocity references (red) from the motion planner and the corresponding measured quantities throughout the experiment. The portion of the data set in Fig. 5 is indicated by the blue dashed lines.

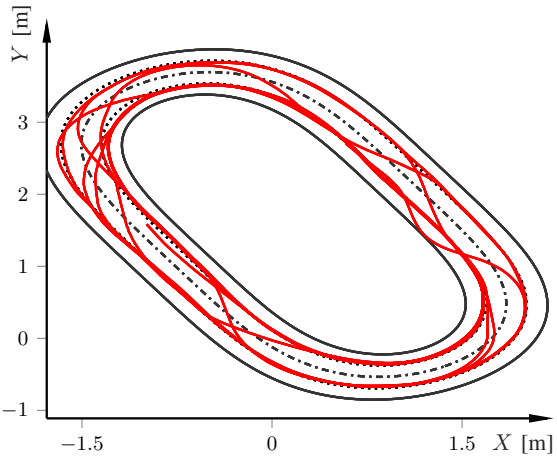

Fig. 7. Measured path for the entire experiment.

in Fig. 6, can reliably track the trajectories generated by the motion planner.

VI. Conclusion

We provided a tutorial overview of the design, implementation, and evaluation of parts of the control stack in autonomous vehicles. Scaled vehicles can be used for testing and verifying the interplay between the different control layers, and also to evaluate the interaction with other parts of the autonomous vehicle software and hardware stack. We demonstrated that scaled vehicle testing can help with avoiding some of the design issues before deployment, thereby shortening the time needed for full-scale testing, with the subsequent implications on cost, time, and safety.

References

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

[2] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior:

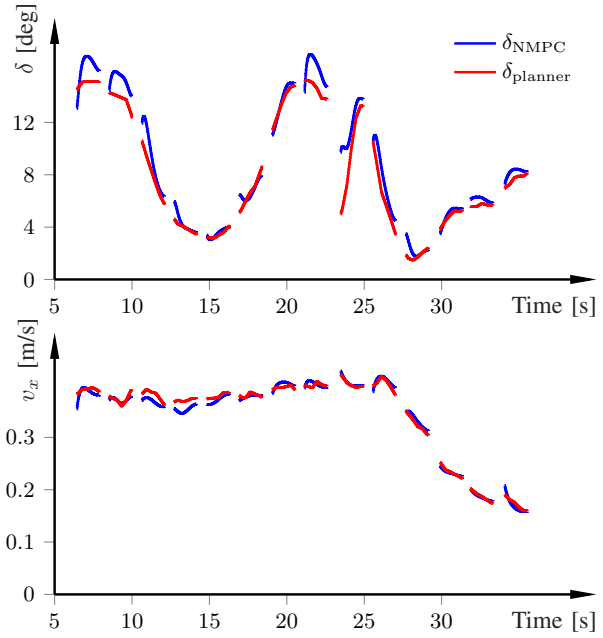

Fig. 8. Steering and velocity references from the motion planner and the corresponding predicted quantities from the NMPC, for 14 planning steps.

The Stanford entry in the Urban challenge," *J. Field R.*, vol. 25, no. 9, pp. 569–597, 2008.

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

[4] S. M. Broek, E. van Nunen, and H. Zwijnenberg, "Definition of necessary vehicle and infrastructure systems for automated driving," Eur. Commission, Tech. Rep. 2010/0064, June 2011.

[5] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[6] ——, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[7] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *J. Field R.*, vol. 25, no. 10, pp. 727–774, 2008.

[8] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[9] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Syst. Mag.*, vol. 36, no. 6, pp. 45–64, 2016.

[10] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *Proceedings 56th Conference on Decision and Control*, 2017.

[11] C. You and P. Tsiotras, "Vehicle modeling and parameter estimation using adaptive limited memory joint-state UKF," in *Amer. Control Conf.*, Seattle, WA, May 2017.

[12] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles," in *Conf. Towards Autonomous Robotic Syst.* Springer, 2013, pp. 276–287.

[13] K. Berntorp and S. Di Cairano, "Tire-stiffness and vehicle-state estimation based on noise-adaptive particle filtering," *IEEE Trans. Control Syst. Technol.*, 2018, in press.

[14] ——, "Particle Gibbs with ancestor sampling for identification of tire-friction parameters," in *IFAC World Congress*, Toulouse, France, July 2017.

[15] ——, "Particle filtering for online motion planning with task specifications," in *Amer. Control Conf.*, Boston, MA, July 2016.

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

[17] O. Arslan, K. Berntorp, and P. Tsiotras, "Sampling-based algorithms for optimal motion planning using closed-loop prediction," in *Int. Conf. Robotics and Automation*, Singapore, May 2017.

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

[19] R. Quirynen, K. Berntorp, and S. Di Cairano, "Embedded optimization algorithms for steering in autonomous vehicles based on nonlinear model predictive control," in *Amer. Control Conf.*, Milwaukee, WI, June 2018.

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

[21] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice–a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[22] R. Quirynen, A. Knyazev, and S. D. Cairano, "PRESAS: Block structured preconditioning within active-set based real-time optimal control," in *Eur. Control Conf.*, vol. (submitted), 2018.

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

[24] Optitrack, "Prime 13 motion capture," 2018, [accessed 23-January-2018]. [Online]. Available: http://optitrack.com/products/prime-13

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

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

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

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

[29] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 299–310, 2009.

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

[31] K. Okamoto, K. Berntorp, and S. Di Cairano, "Similarity-based vehicle-motion prediction," in *Amer. Control Conf.*, Seattle, WA, May 2017.

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

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

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

[35] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V. Kolmanovsky, "The development of model predictive control in automotive industry: A survey," in *Int. Conf. Control Applications*, Dubrovnik, Croatia, 2012.

[36] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, M. Raimondo, and F. Allgöwer, Eds. Springer, 2009, vol. 384, pp. 391–417.

[37] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen, "Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres," *Veh. Syst. Dyn.*, vol. 52, no. 10, pp. 1304–1332, 2014.

[38] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *IFAC World Congress*, Budapest, Hungary, 1984.

[39] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators," *Optimal Control Applications and Methods*, vol. 36, pp. 685–704, 2014.

[40] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of the real-time iteration scheme for nonlinear model predictive control," *IEE Proc.-Control Theory Appl.*, vol. 152, no. 3, pp. 296–308, 2005.