2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

Reinforcement Learning-based Path Following Control for a Vehicle with Variable Delay in the Drivetrain

Johannes Ultsch, Jonas Mirwald, Jonathan Brembeck, Ricardo de Castro

Abstract—**In this contribution we propose a reinforcement learning-based controller able to solve the path following problem for vehicles with significant delay in the drivetrain. To efficiently train the controller, a control-oriented simulation model for a vehicle with combustion engine, automatic gear box and hydraulic brake system has been developed. In addition, to enhance the reinforcement learning-based controller, we have incorporated preview information in the feedback state to better deal with the delays. We present our approach of designing a reward function which enables the reinforcement learning-based controller to solve the problem. The controller is trained using the Soft Actor-Critic algorithm by incorporating the developed simulation model. Finally, the performance and robustness is evaluated in simulation. Our controller is able to follow an unseen path and is robust against variations in the vehicle parameters, in our case an additional payload.**

I. Introduction

Path following control (PFC) - where steering, brakes and engine torque are manipulated in order to make the vehicle accurately follow a given path - is an important task in autonomous vehicles. In cars, which are powered by a combustion engine and an automatic gearbox, the significant time delays in the system response of the drivetrain pose a major challenge for many control algorithms. In addition to delays caused by the combustion engine, the actuation of the hydraulic brake system induces delays in the deceleration response of the vehicle [1].

Recently reinforcement learning (RL) methods have been used to solve a wide range of complex control problems [2].

One reason for the increased interest in applying RL methods to control problems is due to the ability of RL to generate control laws solely through interaction with the plant. The control law can be trained directly out of interaction with the real world system or by using a simulation model. Training the controller in simulation is a promising approach, because it is fast, scalable, safe and has shown positive results (e.g. [3]). Thus, it is considered in this work.

RL methods are increasingly applied in motion control of aerial, marine and ground vehicles. In [4] RL is used to train a controller which is able to steer under-actuated marine vessels along a predefined curved path and it is demonstrated that the controller performs well even under the influence of unpredictable ocean currents. RL methods have also been used to control the inner attitude control loop of unmanned aerial vehicles. In [5] the authors compare different RL algorithms to traditionally-used PID controllers for attitude control. They find that RL is able to outperform the PID controller and that their RL-based controller achieves on average faster rise times and less overshoot than the PID controller. In [6] RL is used to control a real world car by commanding the steering angle and velocity set-points such that it is able to follow a road. This controller relies on feedback data generated by a monocular camera image, inertial measurement units and a steering angle sensor. After training on a real world car on short road segments the controller is able to follow an unknown road for 500 m without intervention.

In [7] the authors show that combined lateral and longitudinal controllers based on neural networks are able to outperform traditional decoupled control laws for path following of vehicles. The application of RL to path tracking of car-like mobile robots has been studied in [8]. Nevertheless, the applicability of the presented approaches to vehicles with combustion engine and automatic gearbox is limited, since the drivetrain dynamics is not considered explicitly. The authors of [9] investigated the benefit of using preview information in the RL-based longitudinal control of a vehicle. In contrast to that, our work focuses on combined longitudinal and lateral control and the incorporation of significant drivetrain delays.

In this work we extend the RL-based PFC framework [10] to cope with conventionally actuated powertrains. More specifically, while [10] focuses on electric and over-actuated powertrains with wheel-based steering and traction actuation [11], this work concentrates on powertrains with a single internal combustion engine, an automatic gearbox and a hydraulic braking system. These latter architectures are more affordable and common in today's automotive industry, which motivated the extension presented in this work.

The contribution of this paper is twofold:

First, we investigate the application of RL to the automotive PFC problem for vehicles with significant and asymmetric delays in the drivetrain. To account for the delays in the actuation we examine the impact of preview information in the RL feedback state. Additionally, we train the controller on two different paths with different characteristics to improve robustness of the RL-controller.

Secondly, a control-oriented simulation model is developed in order to efficiently train the RL-based controller in simulation. The planar vehicle model includes significant asymmetric delays as well as actuation constraints. Moreover, the steering dynamics as well as the hydraulic

The authors are with the Institute of System Dynamics and Control, German Aerospace Center (DLR), Münchener Str. 20, 82234 Wessling, Germany. johannes.ultsch@dlr.de

brake system are approximated by computationally efficient transfer functions.

The remainder of the paper is organized as follows. In section III the path following problem is briefly summarized. Section IV describes the general RL setting and gives a detailed insight into the model used for training. Moreover, the application of RL to the path following problem is discussed. Subsequent the training, the performance as well as the robustness of the obtained controllers is assessed in simulation and presented in section V. The paper finishes with a brief discussion and outlook in section VI.

II. Notation

In this contribution the following notation is used. We denote the reference frame in which the variable is expressed in the superscript. Here, I, P or C correspond to the Inertial, the Path or the Car frame, respectively. The subscript C or P indicates whether the variable is affiliated to the Car or Path.

III. Problem Formulation

The goal is to control the steering angle δ and acceleration set point a_d of a vehicle with combustion engine, hydraulic brake system and front wheel steering such that the vehicle follows a motion demand $\boldsymbol{\lambda}(s)$ parametrized by the arc length s with $\boldsymbol{\lambda}(s) \in \mathbb{R}^5$. The motion demand contains the following values as defined in [12]: the x and y coordinate of the reference path $\boldsymbol{p}_\mathrm{P}^\mathrm{I}(s) = \left[x_\mathrm{P}^\mathrm{I}(s), y_\mathrm{P}^\mathrm{I}(s)\right]^\mathrm{T}$, the corresponding path orientation with respect to the inertial frame $\psi_\mathrm{P}(s)$, the path curvature $\kappa_\mathrm{P}(s)$ and the desired longitudinal velocity tangential to the path $v_d(s)$:

$$\boldsymbol{\lambda}(s) = \left[x_\mathrm{P}^\mathrm{I}(s), y_\mathrm{P}^\mathrm{I}(s), \psi_\mathrm{P}(s), \kappa_\mathrm{P}(s), v_d(s)\right]^\mathrm{T} \tag{1}$$

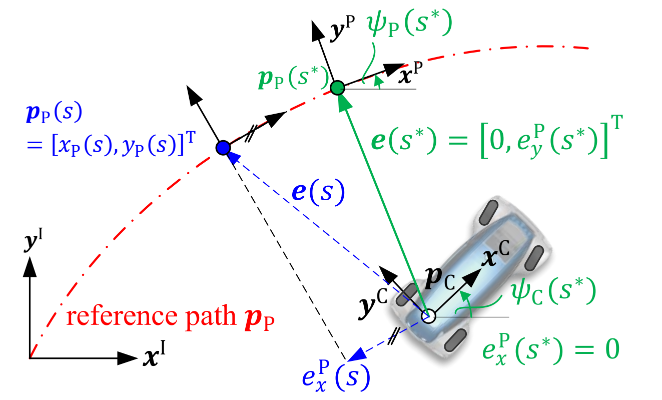
In contrast to trajectory tracking control, in PFC it is not straightforward to determine the reference point on the path used for error calculation. Usually the point on the path which is closest to the vehicle is chosen as reference point:

$$s^* = \arg\min_s \left\| \underbrace{\boldsymbol{p}_\mathrm{P}(s) - \boldsymbol{p}_\mathrm{C}}_{\boldsymbol{e}(s)} \right\|_2 \tag{2}$$

In this work we use the time independent path interpolation (TIPI) from [12] to calculate the arc length s^* which minimizes (2). Incorporating the TIPI has the advantage that no iterative routine is needed to solve equation (2). A solution of the minimization problem (2) exists and is unique if $\boldsymbol{p}_\mathrm{C}$ is closer to the path than the lower bound of the curve radii [13]. The minimization problem (2) as well as the frames of reference are visualized in Fig. 1. The errors which should be minimized by the controller are defined as follows. The cross-track error e_y^P is the difference of the desired lateral position y_P^P and the lateral postion of the car y_C^P, denoted in the path frame. Since in the path frame no lateral offset is desired, this yields $y_\mathrm{P}^\mathrm{P} = 0$ and therefore,

$$e_y^\mathrm{P} = \underbrace{y_\mathrm{P}^\mathrm{P}}_{=0} - y_\mathrm{C}^\mathrm{P}. \tag{3}$$

Similarly, the velocity error $e_{v_x}^\mathrm{P}$ is defined as the difference between the desired velocity v_d in path x direction and the velocity of the car in x direction of the path frame $v_{\mathrm{C},x}^\mathrm{P}$:



$\boldsymbol{y}^\mathrm{P}$
$\psi_\mathrm{P}(s^*)$
$\boldsymbol{p}_\mathrm{P}(s^*)$
$\boldsymbol{x}^\mathrm{P}$
$\boldsymbol{p}_\mathrm{P}(s)$
$= [x_\mathrm{P}(s), y_\mathrm{P}(s)]^\mathrm{T}$
$\boldsymbol{e}(s^*) = [0, e_y^\mathrm{P}(s^*)]^\mathrm{T}$
$\boldsymbol{e}(s)$
$\boldsymbol{y}^\mathrm{I}$
$\boldsymbol{x}^\mathrm{C}$
$\boldsymbol{p}_\mathrm{C}$
$\boldsymbol{y}^\mathrm{C}$
$\psi_\mathrm{C}(s^*)$
reference path $\boldsymbol{p}_\mathrm{P}$
$\boldsymbol{x}^\mathrm{I}$
$e_x^\mathrm{P}(s)$
$e_x^\mathrm{P}(s^*) = 0$

Fig. 1: Visualization of the optimization problem to find s^* (cf. [25])

$$e_{v_x}^\mathrm{P} = v_d - v_{\mathrm{C},x}^\mathrm{P}. \tag{4}$$

Furthermore, we define the orientation error e_ψ to be the angle between the path frame and the car frame:

$$e_\psi = \psi_\mathrm{P} - \psi_\mathrm{C}. \tag{5}$$

Additionally, the acceleration error $e_{a_x}^\mathrm{P}$ and lateral velocity error $e_{v_y}^\mathrm{P}$ are defined as

$$\begin{aligned} e_{a_x}^\mathrm{P} &= a_d - a_{\mathrm{C},x}^\mathrm{P} \\ e_{v_y}^\mathrm{P} &= \underbrace{v_{\mathrm{P},y}^\mathrm{P}}_{=0} - v_{\mathrm{C},y}^\mathrm{P}. \end{aligned} \tag{6}$$

The top priority in our PFC approach is to minimize the cross-track error, because high position errors are likely to cause collisions, e.g. by hitting opposing traffic or by leaving the road. Tracking the velocity reference v_d is ranked as a secondary goal, even though accurate velocity tracking is necessary in order to keep accurate distance to preceding vehicles and stop at traffic lights. When no sideslip is present, minimizing the orientation error is achieved by tracking the reference position. Since we assume Ackermann steering, the instantaneous center of rotation moves along the rear axle in case of no sideslip. Therefore, we take the center of the rear axle as reference point of the vehicle.

IV. Reinforcement Learning Based Path Following Control

In this work we use reinforcement learning to solve the control problem defined in section III. Recently it has become more and more popular to use RL in motion control, as presented in [6], [8] and [9]. Even though ensuring stability of RL-based control approaches is still subject to active research, RL provides advantages also for low-level control compared to traditional control methods. In *model-free* RL, a (near) optimal control law is adapted out of interaction between a RL *agent* and the system that needs to be controlled, the so-called *environment* (cf. Fig. 2).

Especially for complex plants, training the control law out of interaction with a simulation model is appealing, as constructing the control law by hand is avoided. Moreover, multiphysical models created by powerful modeling tools, such as Dymola/Modelica or Simulink/Simscape can be used to train the controller. Therefore, no analytic controller

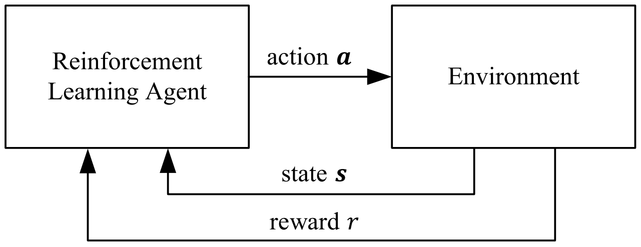


Fig. 2: Reinforcement learning agent environment interaction (cf. [14])

synthesis model of the system needs to be derived as usually needed when applying traditional control methods. Since RL methods are a very generic method, they are not limited to a certain system class (e.g. linear) and the training process can be automatized to a certain degree. Since the optimization is done during training, a lot of the computational effort is shifted to the controller design stage, where high computational power is available. Compared to nonlinear model predictive control, the deployed controller is computationally cheap, since no complex minimization problem has to be solved online.

To explore the state and action space, each RL algorithm exhibits certain randomness during training. This might compromise safety and therefore the agent is trained usually offline in a first stage. In the second stage, the agent can be re-trained with a high fidelity vehicle model to further improve performance.

A. *The Reinforcement Learning Setting*

RL algorithms are developed assuming that the environment can be described as a Markov decision process (MDP) with states $\boldsymbol{s} \in \mathcal{S} \subset \mathbb{R}^n$, actions $\boldsymbol{a} \in \mathcal{A} \subset \mathbb{R}^m$ and transition probability density $p: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)$, which denotes the probability density of transitioning from state $\boldsymbol{s}_k$ at time step k to state $\boldsymbol{s}_{k+1}$ at time step $k+1$ by taking the action $\boldsymbol{a}_k$ [14]. Similar to optimal control, the goal in RL is now to find an optimal policy $\pi^*(\boldsymbol{a}_k|\boldsymbol{s}_k)$ which maximizes the expected sum of rewards $r(\boldsymbol{s}_k, \boldsymbol{a}_k)$:

$$G = \sum_k \mathbb{E}_{(s_k, a_k) \sim \rho_\pi} [r(\boldsymbol{s}_k, \boldsymbol{a}_k)] \tag{7}$$

In this formula the state-action marginal of the trajectory distribution induced by the stochastic policy $\pi(\boldsymbol{a}_k|\boldsymbol{s}_k)$ is denoted by $\rho_\pi(\boldsymbol{s}_k, \boldsymbol{a}_k)$ and the expectation is denominated by $\mathbb{E}$ (cf. [15]). The stochasticity during the training is necessary to discover the full state and action space (exploration), while the main focus of the algorithm is to optimize the policy (exploitation) based on the observed transition trajectories $\boldsymbol{s}_k, \boldsymbol{a}_k, r_{k+1}, \boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}, r_{k+2}, \ldots$. The dilemma to satisfy exploration and exploitation simultaneously is tackled differently by different RL algorithms. In the Soft Actor-Critic (SAC) algorithm [15] this dilemma is tackled by augmenting the standard RL objective in equation (7) by an information-theoretical entropy term $\mathcal{H}(X)$ yielding the optimization objective (cf. [16]):

$$G = \sum_k \mathbb{E}_{(s_k, a_k) \sim \rho_\pi} [r(\boldsymbol{s}_k, \boldsymbol{a}_k) + \alpha \mathcal{H}(\pi(\cdot\,|\boldsymbol{s}_k)] \tag{8}$$

The temperature parameter α in this equation is a trade-off parameter and therefore controls the stochasticity of the policy. A recently proposed enhanced version of the SAC algorithm implements an automatic adaptation of the temperature parameter during training [17]. In this work we chose the enhanced SAC algorithm because it has been shown that it provides sample-efficient learning and at least similar performance in comparison to other widely used RL algorithms [17]. The SAC algorithm makes use of function approximators to learn two soft action-value functions as well as the parameters of the Gaussian policy. The implementation of the SAC algorithm used in this contribution incorporates artificial neural networks as function approximators [18].

B. *Training Model*

The aim of the training model is to reproduce the real world behavior of the system while also providing a short time for simulation. During the training phase, the model is simulated for a couple of hundred thousand time steps. For training a performant controller, several training repetitions are necessary to find the best reward function and hyperparametrization. Because of that, the simulation time of the model determines the time for the development of a RL-based controller essentially.

To keep simulation times limited, we decided to assemble our training model as a planar two-track model with mass m and rotational inertia J. The modeling was done in Dymola/ Modelica using the planar mechanics library [19]. As tire model the "*DryFrictionWheelJoint*" from the planar mechanics library is used, which implements a slip-based dry-friction characteristics.

An overview of the modeling approach for the actuation of the vehicle is given in Fig. 3. The lateral dynamics of the vehicle can be manipulated by commanding an average front axle steering angle δ. To limit the change in the steering angle, a clipping of the commanded steering angle at the wheel is used:

$$\delta_c(k) = \begin{cases} \delta(k), \text{ if } |\delta(k) - \delta_c(k-1)| \leq \Delta\delta_m \\ \delta(k) + \Delta\delta_m, \text{ if } \delta(k) - \delta_c(k-1) > \Delta\delta_m \\ \delta(k) - \Delta\delta_m, \text{ if } \delta(k) - \delta_c(k-1) < -\Delta\delta_m \end{cases} \tag{9}$$

The maximum change in the steering angle during one time step is set to $\Delta\delta_m$ (cf. TABLE 1). Additionally, the commanded steering angle δ is limited to the maximum achievable steering angle (cf. Fig. 3). A low level controller then actuates an electromotor such that the desired steering angle of the by-wire steering system is tracked. The dynamics of the inner steering control loop is modeled as a fixed time delay τ_s followed by a second order transfer function block with damping D_s and angular frequency ω_s.

To control the longitudinal motion of the car, the controller can command a desired acceleration a_d. This commanded acceleration is then limited in the model. In case $a_d > 0$ an positive acceleration is required and only the powertrain is actuated by an allocator (see Fig. 3). $a_d < 0$ corresponds to deceleration and only the braking system is activated. If a positive acceleration a_d is commanded, the acceleration

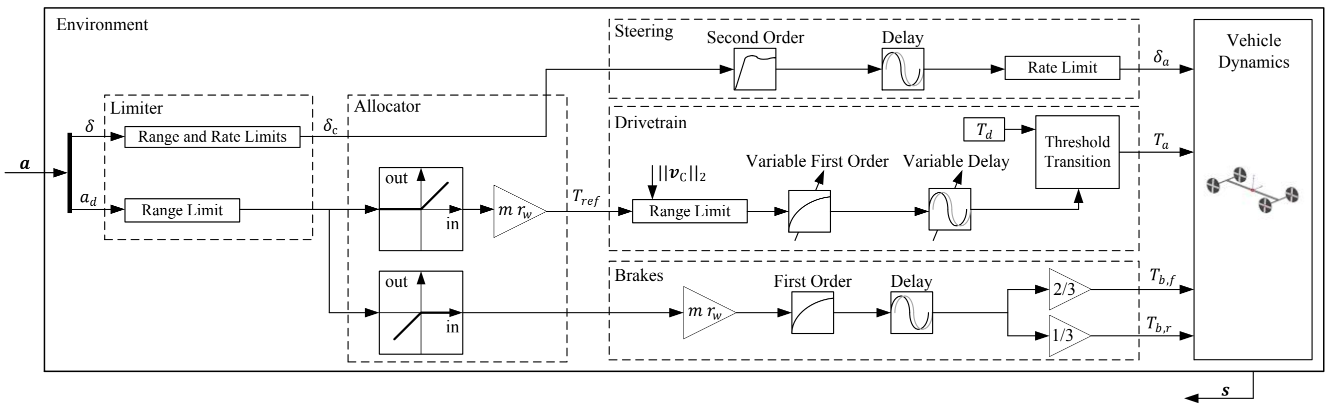


Fig. 3: Block diagram of the environment

reference is converted into an axle torque reference T_{ref} via $T_{ref} = m \cdot r_w \cdot a_d$, neglecting the inertia of the wheels.

To avoid extensive modeling and to keep the simulation model efficient, we assume the powertrain dynamics as a time delay block and a first order transfer function. Usually, the drivetrain exhibits different dynamics when the engine torque rises compared to a decreasing torque. Because of this, the delay time τ and the time constant T of the first order transfer function are switched. To avoid numerical differentiation of the reference torque T_{ref}, the parameters are changed based on whether the reference torque T_{ref} is bigger or smaller than the currently applied torque T_a:

$$\tau = \begin{cases} \tau_r & \text{if } T_{ref} > T_a \\ \tau_f & \text{else} \end{cases} \text{ and } \mathrm{T} = \begin{cases} \mathrm{T}_r & \text{if } T_{ref} > T_a \\ \mathrm{T}_f & \text{else} \end{cases}. \tag{10}$$

The time delay for the rising system dynamics is denoted by τ_r and the time constant for the first order transfer function by T_r. For the falling dynamics the values are denoted by τ_f and T_f, respectively. To avoid high frequency switching between the falling and rising mode in case the reference torque is similar to the applied torque, a hysteresis block is introduced into the switching logic. The drag torque of the engine and gearbox is assumed to be constant and is applied when the engine torque is below a certain fixed threshold. We assume the drag torque at the axle as T_d. The maximum wheel torque the motor can generate is approximated by the envelope of the maximum motor moment from [20] and the gear transmission ratio from [21]. Additionally, a transmission ratio of 3.1 was assumed for the differential. The envelope is included in the drivetrain model as a velocity dependent torque limiter.

Motivated by [1], the dynamics of the hydraulic brake system is modeled as a fixed time delay τ_b and a first order transfer function with time constant T_b (cf. Fig. 3). Similar to the modeling of the drivetrain the commanded acceleration is converted into a brake torque. The obtained torque is then split between the front axle $T_{b,f}$ and the rear axle $T_{b,r}$.

Fig. 4. shows the longitudinal dynamics of the vehicle subject to steps in the commanded acceleration, while keeping the steering input $\delta = 0$. The model parameters are summarized in TABLE 1.

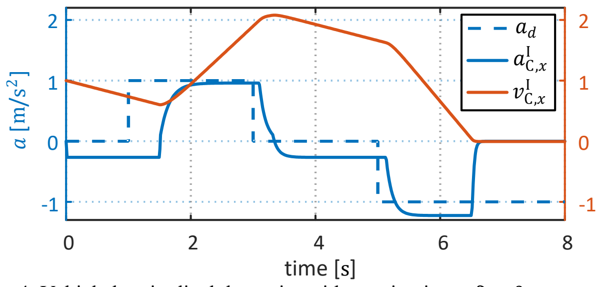


Fig. 4: Vehicle longitudinal dynamics with steering input $\delta = 0$

C. Feedback State

The two major degrees of freedom when designing a RL-based controller are the choice of the feedback state and the reward function. The feedback state should be chosen such that the agent is able to select an appropriate control action based on the information in the feedback state. In our approach, we assembled the feedback state at time step k with input vector $\boldsymbol{a} = [\delta, a_d]^{\mathrm{T}}$ as

$$\boldsymbol{s}(k) = \big[e^{\mathrm{P}}_y(k), e^{\mathrm{P}}_{v_x}(k), e^{\mathrm{P}}_{v_y}(k), e_\psi(k), \kappa_{\mathrm{P}}(k), e^{\mathrm{P}}_{a_x}(k) \\ \hat{e}_\psi(k), \hat{e}^{\mathrm{P}}_{v_x}(k), \delta_{\mathrm{c}}(k), a_d(k), \boldsymbol{s}^{\mathrm{T}}(k-1)\big]^{\mathrm{T}} \tag{11}$$

The feedback state can be divided into different groups: The first group contains the current errors ($e^{\mathrm{P}}_y, e^{\mathrm{P}}_{v_x}, e^{\mathrm{P}}_{v_y}, e_\psi, e^{\mathrm{P}}_{a_x}$), and curvature κ_{P}, the second group contains the preview errors ($\hat{e}_\psi, \hat{e}^{\mathrm{P}}_{v_x}$), the third group contains the clipped steering input δ_{c} and acceleration input a_d and the fourth group is the feedback state from the last time step $\boldsymbol{s}^{\mathrm{T}}(k-1)$.

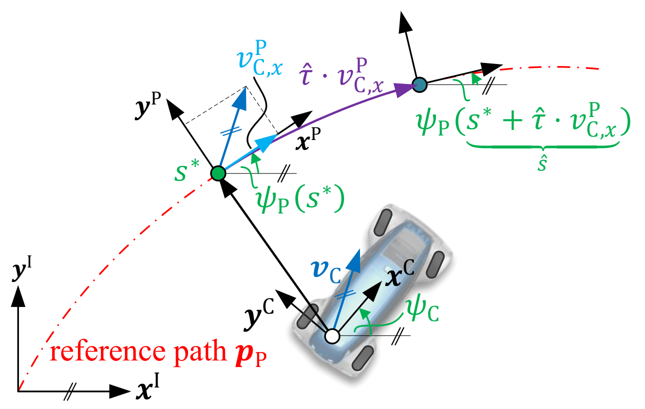
To take the delays of the vehicle into account, the preview errors $\hat{e}_\psi$ and $\hat{e}^{\mathrm{P}}_{v_x}$ are introduced as follows:

$$\hat{e}_\psi(k) = \psi_{\mathrm{P}}(s^*(k) + \hat{\tau}_\psi \cdot v^{\mathrm{P}}_{\mathrm{C},x}(k)) - \psi_{\mathrm{C}}(k) \\ \hat{e}^{\mathrm{P}}_{v_x}(k) = v_d(s^*(k) + \hat{\tau}_{v_x} \cdot v^{\mathrm{P}}_{\mathrm{C},x}(k)) - v^{\mathrm{P}}_{\mathrm{C},x}(k). \tag{12}$$

The calculation of the preview arc length $\hat{s} = s^* + \hat{\tau} \cdot v^{\mathrm{P}}_{\mathrm{C},x}$

TABLE 1: Model parameters

Quantity	Values	Quantity	Values
τ_s	0.05 s	τ_b	0.1 s
D_s	0.5	T_b	0.1 s
ω_s	40 s^{-1}	m	1400 kg
τ_r	0.5 s	J	2000 kg m^2
T_r	0.15 s	T_d	120 Nm
τ_f	0.1 s	r_w	0.31 m
T_f	0.1 s	$\Delta\delta_{\mathrm{m}}$	0.94°



$v_{C,x}^{P}$
$\hat{\tau} \cdot v_{C,x}^{P}$
y^{P}
x^{P}
s^*
$\psi_P(s^* + \hat{\tau} \cdot v_{C,x}^{P})$
$\hat{s}$
$\psi_P(s^*)$
y^I
v_C
x^C
y^C
ψ_C
reference path $\boldsymbol{p}_P$
x^I

Fig. 5: Visualization of the preview calculation

for preview time $\hat{\tau}$ is depicted in Fig. 5. Since the time delay in the steering dynamics is much smaller than the time delay in the longitudinal dynamics, two different preview times are necessary. The preview time for calculating the preview orientation error $\hat{e}_\psi$ is chosen as $\hat{\tau}_\psi = \tau_s$ and the preview for calculating the preview velocity error $\hat{e}_{v_x}$ as $\hat{\tau}_{v_x} = \tau_r$.

D. *Reward Function*

The reward function should be constructed such that maximizing the reward yields good control performance. Since the main control goal is to minimize the cross-track error e_y^P, the velocity error $e_{v_x}^P$ as well as the orientation error e_ψ, we chose the reward r_{PFC} function similar to [10] as

$$r_{\text{PFC}} = r_e + r_{\Delta\delta} + r_{\Delta a_d}. \tag{13}$$

The first term (r_e) promotes error minimization, while $r_{\Delta\delta}$ and $r_{\Delta a_d}$ penalize high changes in the input to achieve a favorable smooth reference. To construct the reward function we make use of two auxiliary functions. The first is a Gaussian-like function $g_{\boldsymbol{\theta}}(x)$, which is defined as

$$g_{\boldsymbol{\theta}}(x) = \theta_1 \, \mathrm{e}^{\frac{-x^2}{2\theta_2}} \tag{14}$$

with $\theta_1, \theta_2 > 0$, yielding $0 < g_{\boldsymbol{\theta}}(x) \leq \theta_1$. The second is a dead zone-based function $h_{\boldsymbol{\beta}}(x)$:

$$h_{\boldsymbol{\beta}}(x) = \begin{cases} 0, & \text{if } |x| < \beta_1 \\ -\beta_2 \cdot |x|, & \text{else} \end{cases}. \tag{15}$$

Since we assume $\beta_1, \beta_2 > 0$, it can be seen that $h_{\boldsymbol{\beta}}(x) \leq 0$. Based on these auxiliary functions, the tracking error term is defined as

$$r_e(e_y^P, e_\psi^P, e_{v_x}^P) := g_{\boldsymbol{\theta}_e}(e_y^P)\Big(1 + g_{\boldsymbol{\theta}_\psi}(e_\psi^P) + g_{\boldsymbol{\theta}_v}(e_{v_x}^P)\Big). \tag{16}$$

To implement the superior control goal, which is the minimization of the cross-track error e_y^P, (16) includes the factor $g_{\boldsymbol{\theta}}(e_y^P)$ on the right-hand side. Since $g_{\boldsymbol{\theta}}(e_y^P)$ tends to zero when the cross-track error tends to infinity, r_e also tends to zero regardless which value the velocity error $e_{v_x}^P$ and the orientation error e_ψ have. This corresponds to our intuition that a high cross-track error yields a low reward, even if the vehicle might be aligned with the path orientation, i.e. $e_\psi = 0$.

To promote favorable steering behavior $r_{\Delta\delta}$ is defined as

$$r_{\Delta\delta}(e_y^P, \Delta\delta) := g_{\boldsymbol{\theta}_{\Delta\delta}}(e_y^P)\, h_{\boldsymbol{\beta}_{\Delta\delta}}(\Delta\delta), \tag{17}$$

with $\Delta\delta(k) = \delta(k) - \delta_c(k-1)$. The factor $g_{\boldsymbol{\theta}}(e_y^P)$ in equation (17) reduces the penalization of high changes in the commanded steering in case of high cross-track errors. Considering that the top goal is to reduce the cross-track error, this implementation seems appealing. High steps in the commanded steering affect driving comfort, which is acceptable in case of high path deviations that might cause severe accidents.

The reward term $r_{\Delta a_d}$ is introduced to minimize jittering in the commanded acceleration a_d and is set to

$$r_{\Delta a_d}(\Delta a_d) := h_{\boldsymbol{\beta}_{\Delta a}}(\Delta a_d) \tag{18}$$

using $\Delta a_d(k) = a_d(k) - a_d(k-1)$. Similarly to (17) $r_{\Delta a_d}$ penalizes high steps in the commanded acceleration a_d. Contrary to (17) it has been observed that including the Gaussian of the cross-track error $g_{\boldsymbol{\theta}}(e_y^P)$ as a factor on the right hand side of (18) yields reduced control performance.

From the definition of the reward function (13)-(18) the maximum value of the reward function is given as

$$r_{\max} = \max\big(r_{\text{PFC}}(\cdot)\big) = \theta_{e,1}\big(1 + \theta_{\psi,1} + \theta_{v,1}\big). \tag{19}$$

E. *Training Setup*

The model described in section IV.B as well as the TIPI have been implemented in Dymola/Modelica and exported as Functional Mock-up Unit (FMU) [22]. Afterwards, the FMUs have been included in a Python framework to be able

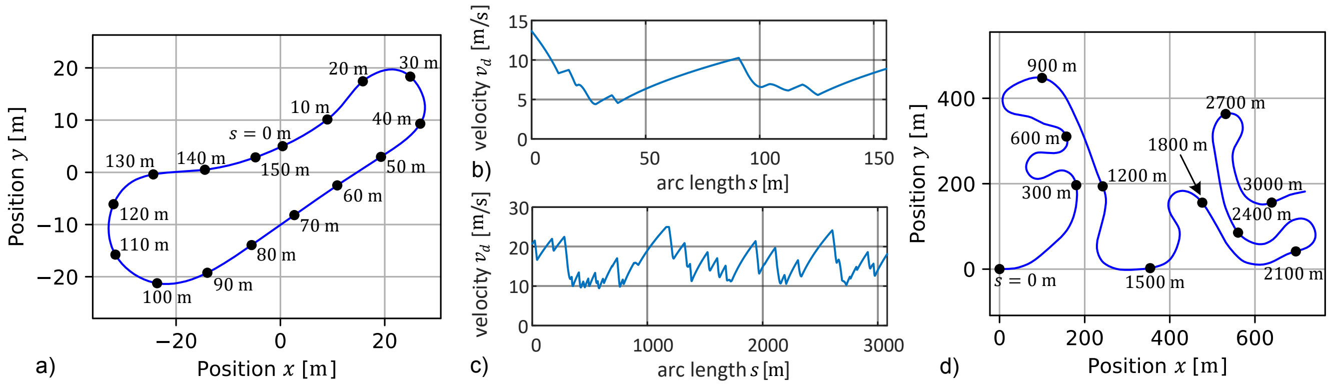


Fig. 6: a) Oval path used for training, b) Velocity demand v_d on oval path, c) Velocity demand v_d on curvy path, d) Curvy path used for training

TABLE 2: Reward function parameters

Parameter values	Parameter values
$\boldsymbol{\theta}_e = [2.0, 0.1]^{\mathrm{T}}$	$\boldsymbol{\theta}_{\Delta\delta} = [1.0, 0.1]^{\mathrm{T}}$
$\boldsymbol{\theta}_\psi = [0.5, 0.005]^{\mathrm{T}}$	$\boldsymbol{\beta}_{\Delta\delta} = [0.0164, 30.6]^{\mathrm{T}}$
$\boldsymbol{\theta}_v = [2.0, 0.2]^{\mathrm{T}}$	$\boldsymbol{\beta}_{\Delta a} = [0.25, 2.0]^{\mathrm{T}}$

to use state-of-the-art RL toolboxes, such as stable-baselines [18]. The planning of the path and velocity profile is out of the scope of this work. We assume that a planning module similar to [23] exists and is able to provide the motion demand (1) as a lookup table. In the real world application the vehicle states $\boldsymbol{p}_{\mathrm{C}}, \boldsymbol{v}_{\mathrm{C}}, \psi_{\mathrm{C}}$ can be estimated e.g. as proposed in [24].

V. Simulative Assessment

After training the RL-based PFC with different hyperparameters and reward functions, the agent which minimizes the maximum cross-track error on the training paths is selected for further evaluation. To assess the benefit of incorporating the preview error terms $\hat{e}_\psi$ and $\hat{e}^{\mathrm{P}}_{v_x}$ in the feedback state $\boldsymbol{s}$, a second controller has been trained with the exact same setting, but without the preview errors in the feedback state $\boldsymbol{s}$. In the following, the controller containing the preview errors is called preRL-PFC and the one without the preview terms RL-PFC.

A. *Training*

The best result with respect to the maximum cross-track error is achieved with the reward function described in section IV.D with the reward function parameters summarized in TABLE 2, trained for 400000 time steps.
It has been observed that initializing the vehicle with a slight offset, orientation and velocity error, which is chosen randomly within a certain range, helps to enhance the robustness of the resulting controller. During the training of the two controller versions (RL-PFC and preRL-PFC) the path is alternated between an oval and a curvy path (see Fig. 6). This way the agent is exposed to different path characteristics which further supports robustness.
The training is assembled out of so-called episodes. An episode is defined as the simulation between initialization of the problem and termination of the simulation. In our PFC setup two kinds of cases, which cause termination, can occur: The first termination case is implemented to avoid exploration of irrelevant regions. Therefore, an episode is aborted when the errors exceed the following limits: $|e^{\mathrm{P}}_y| > 4\ \mathrm{m}$, $|e_\psi| > 80°$, $|e^{\mathrm{P}}_{v_x}| > 5\ \frac{\mathrm{m}}{\mathrm{s}}$, $|e^{\mathrm{P}}_{v_y}| > 5\ \frac{\mathrm{m}}{\mathrm{s}}$, $|\boldsymbol{v}_{\mathrm{C}}| < 1\ \frac{\mathrm{m}}{\mathrm{s}}$. After the abort, a terminal reward $r_{\mathrm{T}} = -3$ is given, the vehicle is reinitialized at the current arc length and the next episode is started. The second case for termination occurs, if a path is completed. After finishing one path, the episode is terminated and the path used for the next episodes is sampled randomly out of two paths. These two paths feature different characteristics: a short oval path with high curvature and a long curvy path, which are both depicted in Fig. 6. Since the oval path can be completed much faster, the oval path is selected with higher frequency to keep the training times on both paths similar. The training is implemented on two paths to expose the controller to different motion demand characteristics and therefore achieve a robust controller.
In Fig. 7 the episode return and episode length of training the preRL-PFC controller is depicted in blue. Moreover, for each episode the maximum reachable return is plotted in red. For an episode with n time steps and maximum reward $r_{\max}$ (see (19)) the maximum reachable return $R_{\max}$ is calculated as

$$R_{\max} = n \cdot r_{\max}\,. \tag{20}$$

It can be seen that after approximately 25000 time steps the episode returns and lengths divide roughly in three branches: A sparse upper branch and two dense branches close to zero, which can be distinguished better in the episode length plot (see Fig. 7). The sparse branch is associated with the training episodes on the long curvy path. Since the curvy path is selected less often, the branch is thinner and because the path is much longer, the collected return is much bigger.
Initializing the vehicle with random initial offset, orientation and velocity error occasionally causes a situation where the agent is not able to avoid violating the abort constraints (e.g. much to large errors), leading to a close-to-zero return and episode length. We have experienced that these challenging initialization conditions help to robustify the controller, even though these early aborts occur during training. The third branch, which is located slightly above the close-to-zero-branch, originates in the training episodes on the oval path.

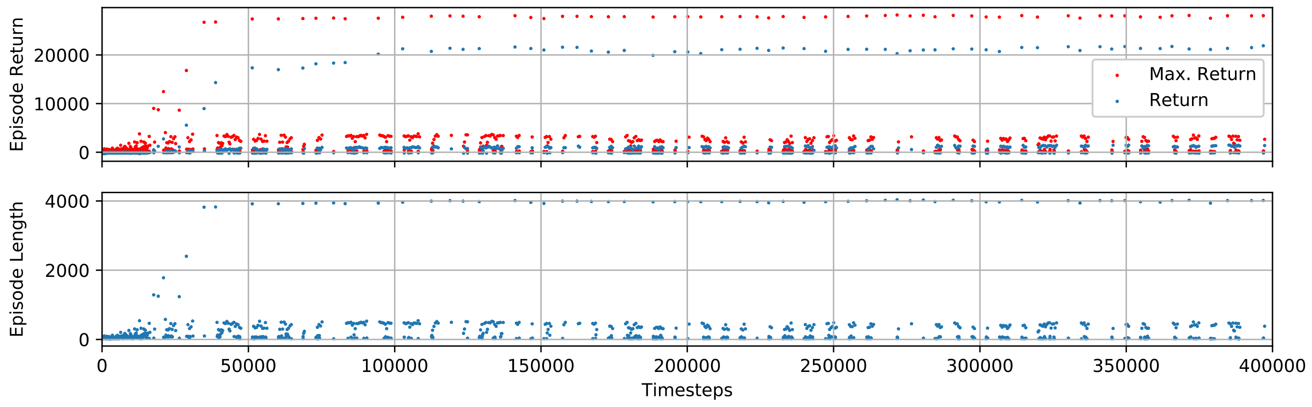


Fig. 7: Episode return, episode length and maximum reachable return during training of the preRL-PFC agent

The episode return plot in Fig. 7 shows, that the maximum reachable return rises just before the obtained returns rise as well. Since the agent first learns to follow the path in a suboptimal way, the episode lengths are increasing which yields an increased maximum reachable return. Due to the fact that the control policy is still suboptimal, the obtained returns are still low. As the agent learns to optimize the policy, the obtained reward rises as well. After the initial step in reward the agent is only able to optimize the reward slightly more.

B. Evaluation on Training Paths

To evaluate the performance of the trained controllers, the agents have been executed on the two paths used for training. The obtained errors are summarized in TABLE 3. It can be seen that including the preview errors offers the agent valuable information, which can be turned into better performance on almost all metrics. Only the velocity errors on the oval path are greater in case of the preRL-PFC agent. Since the first turn of the oval path is very tight (maximum curvature of 0.2) and therefore very challenging to follow, the preRL-PFC learns to reduce the speed in this turn to be able to minimize the cross-track error. Since we defined minimizing the cross-track error to be the main priority, this ends up compromising velocity tracking in favor of a better position tracking.

Even though such extreme driving situations as in the oval path should be avoided in the planning and execution, the agent is able to accurately follow the path.

C. Robustness

To demonstrate the robustness of the learned controllers, the trained agents are also evaluated on an unknown path, the Sachsenring path depicted in Fig. 8. Additionally, we tested the RL-based controllers on the Sachsenring path in a second scenario where we added $\Delta m = 450$ kg to the vehicle mass and applied an additional rotational inertia $\Delta J = 350\,\mathrm{kg\,m^2}$ compared to the vehicle parameters used for training. Since the conversion from acceleration demand a_d to the reference torque would hide the mass deviation from the agent (see Fig. 3), the mass used for conversion is kept unchanged. The resulting errors are displayed in TABLE 4. It can be seen that also on the unknown path the preview error terms helped the preRL-PFC agent to outperform the RL-PFC version of the controller. Since the motion demand of the Sachsenring path is similar to the curvy road, the performance metrics are comparable to the one on the curvy road shown in TABLE 3.

When evaluating the trained controller with and without additional mass and rotational inertia on the Sachsenring path, the agent shows similar performance with respect to the metrics presented in TABLE 4. In comparison to the version with the original mass and inertia, the errors of the preRL-PFC controller are slightly larger (except for $\max(e_\psi)$). Also the evaluation of the RL-PFC controller on the manipulated vehicle model exhibits similar performance as on the nominal model used for training. As can be seen in TABLE 4 the controller is able to compensate the additional mass by applying higher inputs a_d. Since the steering is modeled as pure feedforward, the additional rotational inertia does not affect the steering behavior. Therefore, the applied average steering is nearly the same when evaluating the agent on the vehicle with and without the additional

TABLE 3: Errors when evaluating both trained agent versions on the paths used for training. Both agents were trained alternating on both paths. Listed metrics: the root mean square (RMS), the mean (MEAN) and the maximum (max) of the absolute errors (best metrics on each path are in bold numbers).

Path		Oval		Curvy	
Agent		preRL-PFC	RL-PFC	preRL-PFC	RL-PFC
e_y^{P} [m]	max	**0.45**	1.6	**0.17**	0.41
	RMS	**0.21**	0.63	**0.066**	0.20
$e_{v_x}^{\mathrm{P}}$ [m/s]	max	3.3	**2.3**	**0.72**	1.9
	MEAN	0.81	**0.65**	**0.17**	0.45
e_ψ [°]	max	**5.2**	12	**1.2**	1.8
	MEAN	**1.0**	2.4	**0.20**	0.34

TABLE 4: Errors and inputs when evaluating both trained agents on an unknown path and on an unknown path with additional weight. Both agents were trained alternating on the oval and curvy path. Listed metrics: the root mean square (RMS), the mean (MEAN) and the maximum (max) of the absolute errors (best metrics on each path version are in bold numbers).

Path		Sachsenring		Sachsenring with additional weight	
Agent		preRL-PFC	RL-PFC	preRL-PFC	RL-PFC
e_y^{P} [m]	max	**0.17**	0.40	**0.19**	0.39
	RMS	**0.056**	0.23	**0.060**	0.23
$e_{v_x}^{\mathrm{P}}$ [m/s]	max	**0.68**	2.0	**1.1**	1.9
	MEAN	**0.18**	0.40	**0.19**	0.41
e_ψ [°]	max	3.7	**1.4**	3.5	**1.3**
	MEAN	**0.17**	0.33	**0.18**	0.28
a_d [m/s^2]	MEAN	**1.49**	1.57	**1.80**	1.86
δ_c [°]	MEAN	1.35	**1.34**	1.35	1.35

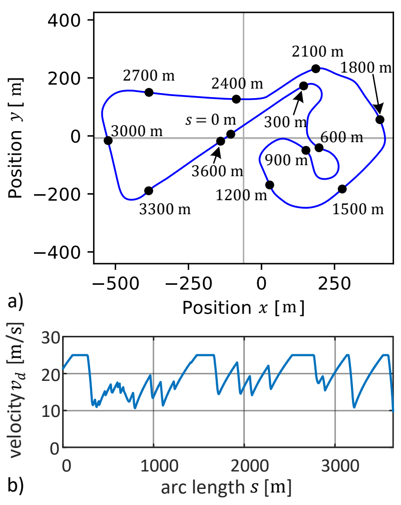


Fig. 8: a) Sachsenring path used for evaluation, b) Velocity demand v_d on Sachsenring path

mass (cf. TABLE 4). This results show the robustness of the obtained controller against changes in the vehicle weight. Considering the significant delays in the vehicle actuation, the errors are in a range which is suitable for application tests on a real world vehicle. The robustness analysis with additional weight and rotational inertia also yields encouraging results looking well for future tests in a real world prototype.

VI. Discussion and Outlook

In this contribution a RL-based controller for solving the path following problem for a vehicle with significant delays in the actuation has been proposed. Additionally, a control-oriented model for a vehicle with combustion engine, automatic gearbox, hydraulic brake system and steer-by-wire suitable for training a RL-based controller has been developed. To minimize the effect of the significant delays in the vehicle model, additional preview error terms have been introduced in the feedback state. The training of the controller is split onto two paths with different characteristics to obtain a more robust controller. Finally, the performance of the controllers has been assessed in simulation, showing robust performance on an unseen path and parameter uncertainty. These promising simulation results encourage us to test the controller on a real world vehicle in future works.

VII. References

[1] M. L. Kuang, M. Fodor, D. Hrovat and M. Tran, "Hydraulic brake system modeling and control for active control of vehicle dynamics," in *Proceedings of the 1999 American Control Conference*, 1999.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations,* 2016.

[3] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," *(arXiv preprint arXiv:1804.10332),* 2018.

[4] A. B. Martinsen and A. M. Lekkas, "Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models," in *OCEANS 2018 MTS/IEEE Charleston*, 2018.

[5] W. Koch, R. Mancuso, R. West and A. Bestavros, "Reinforcement Learning for UAV Attitude Control," *ACM Transactions on Cyber-Physical Systems,* vol. 3, pp. 1-21, 2 2019.

[6] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley and A. Shah, "Learning to Drive in a Day," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[7] G. Devineau, P. Polack, F. Altche and F. Moutarde, "Coupled Longitudinal and Lateral Control of a Vehicle using Deep Learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[8] D. Kamran, J. Zhu and M. Lauer, "Learning Path Tracking for Real Car-like Mobile Robots From Simulation," in *2019 European Conference on Mobile Robots (ECMR)*, 2019.

[9] M. Buechel and A. Knoll, "Deep Reinforcement Learning for Predictive Longitudinal Control of Automated Vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[10] J. Ultsch, J. Brembeck and R. Castro, "Learning-Based Path Following Control for an Over-Actuated Robotic Vehicle," in *VDI-AUTOREG*, Düsseldorf, 2019.

[11] J. Brembeck, L. M. Ho, A. Schaub, C. Satzger, J. Tobolar, J. Bals and G. Hirzinger, "ROMO – The Robotic Electric Vehicle," in *22nd IAVSD International Symposium on Dynamics of Vehicle on Roads and Tracks*, 2011.

[12] P. Ritzer, C. Winter and J. Brembeck, "Advanced path following control of an overactuated robotic vehicle," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2015.

[13] P. Morin and C. Samson, "Motion Control of Wheeled Mobile Robots," in *Springer Handbook of Robotics*, Springer Berlin Heidelberg, 2008, pp. 799-826.

[14] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Second edition ed., The MIT Press, 2018.

[15] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholmsmässan, Stockholm Sweden, 2018.

[16] B. D. Ziebart, "Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy," Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 2010.

[17] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine, "Soft Actor-Critic Algorithms and Applications," *(arXiv preprint arXiv:1812.05905v2),* 2018.

[18] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor and Y. Wu, *Stable Baselines,* https://github.com/hill-a/stable-baselines, 2018.

[19] D. Zimmer, "A Planar Mechanical Library for Teaching Modelica," in *Proceedings of the 9th International MODELICA Conference*, 2012.

[20] C. Schwarz, S. Missy, H. Steyer, B. Durst, E. Schünemann, W. Kern and A. Witt, "Die neuen Vier-und Sechszylinder-Ottomotoren von BMW mit Schichtbrennverfahren," *MTZ – Motortechnische Zeitschrift,* 5 2007.

[21] W. Hinz and G. Bednarek, "Achtgang-Automatikgetriebe für den neuen Opel Insignia," *ATZ - Automobiltechnische Zeitschrift,* vol. 118, pp. 16-21, 11 2016.

[22] Modelica Association, "FMI Standard," [Online]. Available: https://fmi-standard.org/.

[23] C. Winter, P. Ritzer and J. Brembeck, "Experimental investigation of online path planning for electric vehicles," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[24] J. Brembeck, "Nonlinear Constrained Moving Horizon Estimation Applied to Vehicle Position Estimation," *Sensors,* vol. 19, no. 10, p. 2276, 2019.

[25] J. Brembeck, "Model Based Energy Management and State Estimation for the Robotic Electric Vehicle ROboMObil," Ph.D. thesis, Technical University of Munich, Munich, 2018.