Learning-Based Path Following Control for an Over-Actuated Robotic Vehicle

Johannes Ultsch; Dr. Jonathan Brembeck; Dr. Ricardo de Castro,
Institute of System Dynamics and Control, Department for Vehicle System Dynamics, German Aerospace Center, Oberpfaffenhofen

Kurzfassung

Für autonome Fahrzeuge stellt die Pfadfolgeregelung eine Schlüsselfunktion dar. Die Pfadfolgeregelung steuert hierbei Antrieb, Lenkung und Bremse derart, dass das Fahrzeug einem geometrischen Pfad mit einer Referenzgeschwindigkeit folgt. Für die Auslegung von leistungsfähigen modellbasierten Pfadfolgereglern wird ein ausreichend genaues Synthesemodell des Fahrzeuges benötigt. Der Entwurf, die Parametrierung und das Testen von modellbasierten Pfadfolgereglern, sowie das Ableiten eines Synthesemodells ist allerdings eine zeitaufwändige Aufgabe. In der klassischen Regelungstechnik werden deshalb vermehrt Reinforcement Learning (RL) Methoden angewandt, um Regelungsprobleme ohne Synthesemodell, nur mit Hilfe von hochgenauen Simulationsmodellen zu lösen. Um den Einsatz von RL Methoden auf das Pfadfolgeproblem zu untersuchen, wird in diesem Beitrag die Anwendung am Beispiel des überaktuierten robotischen Fahrzeuges RObоMObil des DLRs vorgestellt. Erste Simulationsergebnisse zeigen, dass RL basierte Pfadfolgeregler auf dem Trainingspfad ein ähnlich gutes Folgeverhalten aufweisen, wie modellbasierte Pfadfolgeregler. Die RL basierten Regler erzielen dabei auch auf neuen und unbekannten Pfaden gute Ergebnisse.

Abstract

Motion control, in particular path following control (PFC), is an important function of autonomous vehicles. PFC controls the propulsion, steering and braking such that the vehicle follows a parametric path and reference velocity. For the design of traditional model-based PFC approaches a sufficiently accurate synthesis model of the vehicle has to be available in order to design a performant controller. However, constructing, parametrizing and testing these model-based PFC as well as deriving the synthesis model is known to be a time-consuming task. Recently the application of reinforcement learning (RL) methods to solve control problems without a synthesis model but based on high fidelity simulation models has gained increasing interest. In this paper we investigate the application of RL methods to solve the path following problem for DLR's RObоMObil, an over-actuated robotic

vehicle. Simulation results demonstrate that the RL-based PFC exhibits similar tracking performance as a model-based controller, executed on the path used for training. Moreover the RL-based PFC provides encouraging generalization capabilities, when facing unseen reference paths.

1 Introduction

Besides perception and path planning, autonomous driving requires motion control strategies capable of keeping the vehicle on a reference path or trajectory. In order to safely follow the path the aim of the motion controller is to minimize the displacement between the vehicle and a reference path while tacking a desired velocity. This paper focusses on motion control algorithms for path following problems, where the motion demand is characterized by a geometric path description in dependency of the arc length. In contrast to trajectory tracking control, path following control does not require a time-dependent path description.

In previous works, this kind of control problem has been tackled with model-based control methods, where a kinematic or dynamic model of the vehicle is employed to systematically construct the control law, while fulfilling important control requirements, like input and state constraints as well as closed loop robust stability [1], [2], [3], [4], [5], [6].

Despite achieving good results, model-based control methods require an accurate synthesis model of the vehicle in order to design and tune the controller. In most cases the complexity of the model used in controller design is however limited by the control method used [1]. For example, widely used linear control methods only perform well if the controlled system is linear or close to linear in the points of operation. Model predictive control (MPC) with a high number of states and nonlinearities requires a high computational effort to solve the optimization problem online. Another drawback of model-based control is the high human effort in the design process, which is time consuming, tedious and error prone.

Recently, it has been shown that reinforcement learning (RL) methods are able to solve a wide variety of control problems, ranging from classical cart pole balancing [7] to flight control for unmanned aerial vehicles [8] and path following of marine vessels [9]. In the model-free reinforcement learning setting no synthesis model is necessary because the control policy is adapted based on observed data and manipulating the control inputs of the system. Similar to model predictive control the control objective is formulated as a numerical reward function (cf. cost function in MPC), which the reinforcement learning agent tries to maximize. The procedure of adapting the control policy based on interaction with the plant is called training and can be done offline in simulation or directly online with the real plant [10].

In this work we present two variants of RL-based path following controllers which are able to track unknown paths. Additionally, the learning based controllers are compared to a model based controller from previous works within the ROboMObil project [6], [11].

To facilitate the integration of physical models in the training of the RL algorithm, we developed a highly modular training framework based on Python and Functional Mock-up Units (FMU). The FMUs contain the physical model as compiled code and can be created by different modelling tools like Dymola.

The Functional Mock-up Interface (FMI) [12] is an open standard for exchanging physical models between different simulation and modelling tools.

By incorporating one FMU for the vehicle model and one FMU for calculating the reference point on the path a highly modular training framework is achieved and new vehicle configurations can be adapted easily.

2 Problem Formulation

In this section the vehicle model, the parametric path representation and the algorithm implemented to determine the reference point on the path are presented.

Throughout this work the frame of reference is denoted by a superscript I, C, P or W for the inertial, car, path and wheel frame, respectively. If the frame of reference is not relevant the superscript is omitted for better readability. The distinction between car and path signals is done by subscripts C or P.

2.1 Vehicle Model

The vehicle model incorporated is based on the configuration of DLR's research vehicle ROboMObil [13] (cf. Fig. 1) and consists of a planar two track model including a nonlinear tire model implemented in Modelica using the planar mechanics library [14]. In this setting road height information is neglected and all used reference paths are projected into the xy-plane. The planar mechanics library contains wheel models with different modelling complexity adapted from [15]. The used tire model (*"DryFrictionWheelJoint"*) relies on a semi-empirical formula which considers combined slip and low computational effort. The parameters for the μ-slip characteristics of the used tyre model are obtained from real world test data. In the modelling only static vertical loads are considered.

The vehicle can be controlled by commanding the steering angles δ^{W_i} of each wheel as well as the torque τ^{W_i} of the in-wheel-motors as shown in Fig. 1. The low level control inputs are gathered in $\boldsymbol{u} = [\delta^{\mathrm{W}_1}, \delta^{\mathrm{W}_2}, \delta^{\mathrm{W}_3}, \delta^{\mathrm{W}_4}, \tau^{\mathrm{W}_1}, \tau^{\mathrm{W}_2}, \tau^{\mathrm{W}_3}, \tau^{\mathrm{W}_4}]$ and are used to control the three planar degrees of freedom of the car. Since the dimension of the control inputs exceeds the

degrees of freedom, the control problem is over actuated and a control allocation needs to be applied.

The vehicle velocity $\boldsymbol{v}_{\mathrm{C}}^{\mathrm{C}} = \left[v_{\mathrm{C},x}^{\mathrm{C}}, v_{\mathrm{C},y}^{\mathrm{C}}\right]$ in the car frame of reference as well as the tire forces $\boldsymbol{F}^{\mathrm{W}_i} = [\boldsymbol{F}_x^{\mathrm{W}_i}, \boldsymbol{F}_y^{\mathrm{W}_i}]$ in the wheel frame of reference are also depicted in Fig. 1.

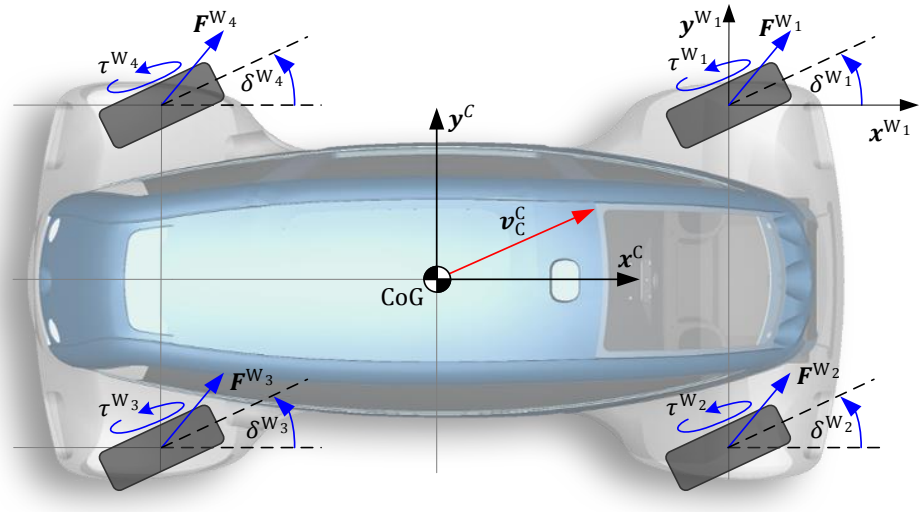

Fig. 1: Vehicle configuration of DLR's ROboMObil

Due to the fast time-response of the in-wheel-motors their dynamics are neglected in the model. The dominant dynamics of the steering actuator is approximated by a second order system (Bessel filter with critical frequency of $15\ \mathrm{Hz}$) and the maximum actuator rate is limited by $\left|\dot{\delta}^{\mathrm{W}_i}\right| \leq 60\ {}^{\circ}/_{s}, i \in \{1, \ldots, 4\}$. All vehicle parameters are summarized in [6].

2.2 The Parametric Path Description

The interface to the path following control is a motion demand parametrized by the path arc length s: $\boldsymbol{\lambda}(s) \in \mathbb{R}^5$. It consists of the following five quantities: the absolute estima $\boldsymbol{p}_{\mathrm{P}}^{\mathrm{I}} = \left[x_{\mathrm{P}}^{\mathrm{I}}(s), y_{\mathrm{P}}^{\mathrm{I}}(s)\right]$ of the reference path, the corresponding path orientation $\psi_{\mathrm{P}}(s)$, its curvature $\kappa_{\mathrm{P}}(s)$ and a desired longitudinal velocity $v_{\mathrm{P},x}^{\mathrm{P}}(s)$:

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

Fig. 2 depicts an example of the motion demand at the point s_i. Note that the desired vehicle longitudinal velocity $v_x^{\mathrm{P}}(s_i)$ is in the direction of the tangent vector $\boldsymbol{t}^{\mathrm{P}}$ at $\boldsymbol{p}_{\mathrm{P}}^{\mathrm{I}}(s_i)$. The normal vector $\boldsymbol{n}^{\mathrm{P}}$ is orthogonal to $\boldsymbol{t}^{\mathrm{P}}$ and spans together with $\boldsymbol{t}^{\mathrm{P}}$ the path frame of reference.

Since path planning is out of the scope of this contribution it is assumed that a parametric path is available in form of a lookup table. An overview of online path planning for the ROboMObil is given in [16].

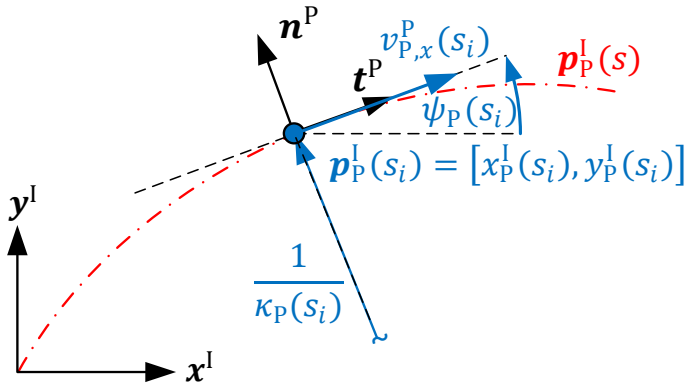

Fig. 2: Graphical interpretation of the parametric path $\boldsymbol{\lambda}(s)$ at point s_i

2.3 Time Independent Path Interpolation (TIPI)

Consider the vehicle position $\boldsymbol{p}_{\mathrm{C}}$ as depicted in Fig. 3. In order to implement the PFC we need to find the closest point between $\boldsymbol{p}_{\mathrm{C}}$ and the reference path $\boldsymbol{p}_{\mathrm{P}}$. In other words, we need to find s where the distance $\boldsymbol{e}(s) = \boldsymbol{p}_{\mathrm{P}}(s) - \boldsymbol{p}_{\mathrm{C}}$ between the path and the position of the car is minimal, i.e.

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

The geometrical interpretation of this minimization objective is that $\boldsymbol{p}_{\mathrm{P}}(s^*)$ can be determined by projecting $\boldsymbol{p}_{\mathrm{C}}$ orthogonally on the path $\boldsymbol{p}_{\mathrm{P}}(s)$ which yields that the x-coordinate of the position error in the path frame is zero: $e^{\mathrm{P}}_x(s^*) = 0$ (cf. Fig. 3).

It is assumed that the initial vehicle position $\boldsymbol{p}^{\mathrm{I}}_{\mathrm{C}}$ is sufficiently accurately estimated by a vehicle position estimator e.g., as proposed [17]. The optimization problem (2) is solved by means of a time independent path interpolation (TIPI), which is adopted from [6], [11].

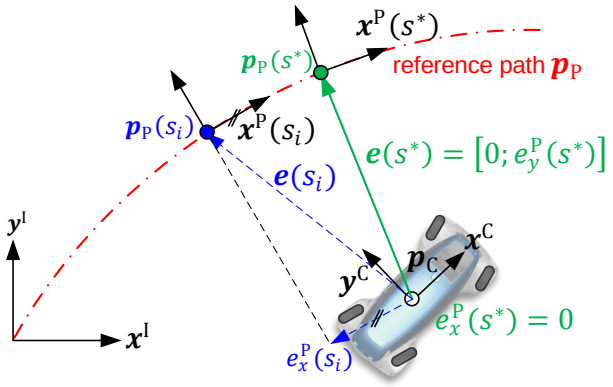

Fig. 3: Graphical representation of the dynamic root finding to determine s^*

As stated in [18] $\boldsymbol{p}_{\mathrm{P}}(s^*)$ exists and is unique if $e^{\mathrm{P}}_y(s^*) \leq 1/\kappa_{\mathrm{P}}(s^*)$. With the origin of the path frame of reference located in $\boldsymbol{p}_{\mathrm{P}}(s^*)$ the errors represented in the path frame, which will be used later in the controller setup, can be calculated as follows:

$$
\begin{aligned}
e_y^{\mathrm{P}} &= \underbrace{y_{\mathrm{P}}^{\mathrm{P}}}_{=0} - y_{\mathrm{C}}^{\mathrm{P}} \\
e_{v_x}^{\mathrm{P}} &= v_{\mathrm{P},x}^{\mathrm{P}} - v_{\mathrm{C},x}^{\mathrm{P}} \\
e_{v_y}^{\mathrm{P}} &= \underbrace{v_{\mathrm{P},y}^{\mathrm{P}}}_{=0} - v_{\mathrm{C},y}^{\mathrm{P}} \\
e_{\psi} &= \psi_{\mathrm{P}} - \psi_{\mathrm{C}}.
\end{aligned}
\tag{3}
$$

3 Reinforcement Learning based Path Following Control

In the beginning of this section a general reinforcement learning setting is described and the chosen RL algorithm is discussed briefly. After this the whole structure of the RL-based PFC is presented and the incorporated interfaces are explained. In the end details on the implementation are given.

3.1 Reinforcement Learning Setting

In the reinforcement learning setting it is assumed that the controlled environment can be described as a Markov decision process (MDP) with a continuous set of states $\boldsymbol{s} \in \mathcal{S}$ and a continuous set of actions $\boldsymbol{a} \in \mathcal{A}$ (cf. [19], [20]). The probability for transitioning from a state $\boldsymbol{s}_t$ at time step t to the state $\boldsymbol{s}_{t+1}$ at time step $t+1$ by taking action $\boldsymbol{a}_t$ is given by a state transition probability $p(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, \boldsymbol{a}_t): \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)$ and a reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t): \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is received at each transition.

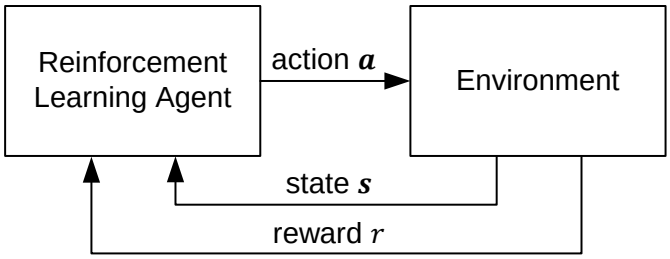

Fig. 4: Reinforcement learning setting

The standard objective in reinforcement learning is to find an optimal stochastic control policy $\pi^*(\boldsymbol{a}_t|\boldsymbol{s}_t)$ such that the expected sum of rewards

$$
G = \sum_t \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} [r(\boldsymbol{s}_t, \boldsymbol{a}_t)] \tag{4}
$$

is maximized with $\rho_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denoting the state-action marginal of the trajectory distribution induced by the stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and expectation $\mathbb{E}$ (cf. [20]). After the training process the stochastic policy is transformed into a deterministic policy by taking the expected value at each state $\boldsymbol{s}_t$.

In *model-free* reinforcement learning it is assumed that the environment is unknown and the policy is learned out of interaction between the agent and the environment as shown in Fig. 4. During training the agent adapts the policy to maximize the reward (exploitation) by still acting random enough to explore for potentially better solutions (exploration).

Recently several algorithms have been published which aim to solve this problem incorporating artificial neural nets [21], [22]. In our work we chose the Soft Actor-Critic (SAC) [20] algorithm because of its stable behavior during training and its robust training over a wide range of hyper parameters. In reinforcement leaning, parameters like the number of time steps considered for training, which parametrize the RL algorithm are called hyper parameters. It has been shown that in addition to that SAC achieves similar performance than other RL algorithms [20].

In SAC the objective to maximize the expected sum of rewards is augmented by an entropy term which considers exploration by design [20]:

$$\pi^* = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\rho_\pi}[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha\mathcal{H}(\pi(\cdot\,|\boldsymbol{s}_t))] \tag{5}$$

with the information-theoretical entropy $\mathcal{H}(X)$ (cf. [23]).

The relative importance between the maximization of the return and the maximization of the entropy is adjusted by the temperature parameter α. In an enhanced version of the SAC, which is used in this contribution, the temperature parameter is selected automatically during training which minimizes the effort of hyper parameter tuning [24].

3.2 Application of Reinforcement Learning to the Path Following Problem

To reduce the problem complexity we split the PFC problem into a path following and a control allocation problem. We then use RL to solve the path following problem.

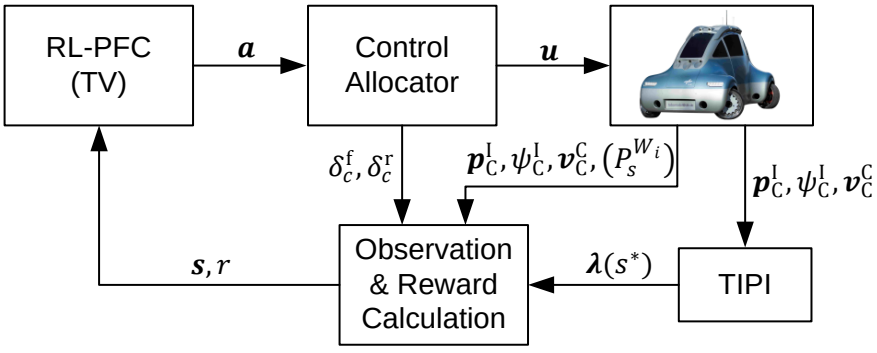

Fig. 5: Reinforcement Leaning based PFC setup

As shown in Fig. 5 a control allocator maps the actions $\boldsymbol{a}$ commanded by the RL algorithm to the vehicle inputs $\boldsymbol{u}$. In order to gain insights in the ability of RL to deal with over actuation

we designed two variants of RL-based PFCs and compare them (cf. section 3.2.1 and 3.2.2). Since RL works in discrete time the system is sampled with sample time $T_s = 0.05\ \mathrm{s}$

3.2.1 Basic Reinforcement Learning based Path Following Control (RL-PFC)

In the first version (RL-PFC) the action space of the RL is selected such that front axle steering angle δ_t^{f}, rear axle steering angle δ_t^{r} and average in-wheel motor torque τ_t can be commanded: $\boldsymbol{a}_t = \left[\delta_t^{\mathrm{f}}, \delta_t^{\mathrm{r}}, \tau_t\right]$. The control allocator then distributes the desired torque equally to all wheels: $\tau_t = \tau_t^{\mathrm{W}_1} = \tau_t^{\mathrm{W}_2} = \tau_t^{\mathrm{W}_3} = \tau_t^{\mathrm{W}_4}$. Since the steering actuators can only realize a certain steering rate $\dot{\delta}_{\max}$ the commanded steering angles δ_t^{f} and δ_t^{r} are constrained by the control allocator in each time step t: $\delta_{c,t}^{i} = f_{\mathrm{clip}}(\delta_t^{i}, \delta_{c,t-1}^{i}, \dot{\delta}_{\max})$ for $i \in \{\mathrm{f}, \mathrm{r}\}$ with

$$f_{\mathrm{clip}}\left(x_t, x_{t-1}, \dot{\delta}_{\max}\right) = \begin{cases} x_t, & \text{if } |x_t - x_{t-1}| \leq T_s\, \dot{\delta}_{\max} \\ x_{t-1} + T_s\, \dot{\delta}_{\max}, & \text{if } x_t - x_{t-1} > T_s\, \dot{\delta}_{\max} \\ x_{t-1} - T_s\, \dot{\delta}_{\max}, & \text{if } x_t - x_{t-1} < -T_s\, \dot{\delta}_{\max}\,. \end{cases} \tag{6}$$

The clipped front and rear axle steering angles $\delta_{c,t}^{\mathrm{f}}$ and $\delta_{c,t}^{\mathrm{r}}$ are then allocated according to the Ackermann steering assumption such that all wheels have the same instantaneous center of rotation (ICR) [25]. For the calculation it is assumed that $\delta_{c,t}^{\mathrm{f}}$ steers a virtual wheel in the middle of the front axle and $\delta_{c,t}^{\mathrm{r}}$ a virtual wheel in the middle of the rear axle. The intersection of the orthogonal lines onto this virtual wheels uniquely define the ICR and the steering angles $\delta_t^{\mathrm{W}_1}, \delta_t^{\mathrm{W}_2}, \delta_t^{\mathrm{W}_3}, \delta_t^{\mathrm{W}_4}$ can be calculated. If $\delta_{c,t}^{\mathrm{f}}$ and $\delta_{c,t}^{\mathrm{r}}$ have the same value, the ICR moves to infinity and this value is applied to each wheel.

3.2.2 Reinforcement Learning based Path Following Control with Torque Vectoring (RL-PFC TV)

To investigate how the RL algorithm deals with over-actuation a second variant is designed (RL-PFC TV). In this variant the action vector $\boldsymbol{a}_t$ is extended with the torque demands of each wheel: $\boldsymbol{a}_t = \left[\delta_t^{\mathrm{f}}, \delta_t^{\mathrm{r}}, \tau_t^{\mathrm{W}_1}, \tau_t^{\mathrm{W}_2}, \tau_t^{\mathrm{W}_3}, \tau_t^{\mathrm{W}_4}\right]$. This gives the RL agent the possibility to utilize torque vectoring (TV) and no control allocation for the wheel torques is necessary. The steering angles are again distributed as in the RL-PFC variant described in section 3.2.1.

3.2.3 Reward Function for Path Following Control

Besides selecting the RL hyper parameters, design choices have to be made by formulating the reward function as well as selecting the observation vector $\boldsymbol{s}$. Similar to the cost function in model predictive control the reward function should be designed such that fulfilling the control goal yields a high scalar reward (cf. low cost in MPC).

In the construction of the reward functions we make use of a dead zone-based function $h_{\boldsymbol{\beta}}(x)$ inspired by [26]

$$h_{\boldsymbol{\beta}}(x) = \begin{cases} 0, & \text{if } |x| < \beta_1 \\ -\beta_2 \cdot |x|, & \text{else} \end{cases} \tag{7}$$

and of a Gaussian-like function $g_{\boldsymbol{\theta}}(x)$ inspired by [9]

$$g_{\boldsymbol{\theta}}(x) = \theta_1 \, e^{\frac{-x^2}{2\theta_2}}. \tag{8}$$

It should be noted that $g_{\boldsymbol{\theta}}(x)$ fulfils

$$0 < g_{\boldsymbol{\theta}}(x) \leq \theta_1, \forall\, x \in \mathbb{R}, \forall\, \theta_1, \theta_2 \in \mathbb{R}^+ \tag{9}$$

which is used later.

The main control goal of the PFC is to keep the vehicle close to the path ($e_y^{\mathrm{P}} \to 0$) because deviation from the demanded path might possibly compromise safety e.g. collisions with obstacles. As secondary goals, we consider the minimization of the orientation error e_ψ and longitudinal velocity error $e_{v_x}^{\mathrm{P}}$.

To achieve this behavior the following reward function, inspired by [9] and [26], is designed for the RL-PFC:

$$r_{\mathrm{RL-PFC}}\left(e_y^{\mathrm{P}}, e_\psi^{\mathrm{P}}, e_{v_x}^{\mathrm{P}}, \Delta\delta^{\mathrm{f}}, \Delta\delta^{\mathrm{r}}\right) = r_e\left(e_y^{\mathrm{P}}, e_\psi^{\mathrm{P}}, e_{v_x}^{\mathrm{P}}\right) + r_{\Delta\delta}\left(e_y^{\mathrm{P}}, \Delta\delta^{\mathrm{f}}, \Delta\delta^{\mathrm{r}}\right). \tag{10}$$

Herein $\Delta\delta_t^i$ denotes the difference between the commanded steering angle and the clipped steering angle from the last time step $t-1$: $\Delta\delta_t^i = \delta_t^i - \delta_{c,t-1}^i$ for $i \in \{\mathrm{f}, \mathrm{r}\}$. The reward function $r_{\mathrm{RL-PFC}}$ is composed of two terms r_e and $r_{\Delta\delta}$. The first one accounts for tracking errors while the second one penalizes control jittering in the steering angles.

The first term r_e is chosen as follows, extended from [9]:

$$r_e\left(e_y^{\mathrm{P}}, e_\psi^{\mathrm{P}}, e_{v_x}^{\mathrm{P}}\right) := g_{\boldsymbol{\theta}_e}\left(e_y^{\mathrm{P}}\right)\left(1 + g_{\boldsymbol{\theta}_\psi}\left(e_\psi^{\mathrm{P}}\right) + g_{\boldsymbol{\theta}_v}\left(e_{v_x}^{\mathrm{P}}\right)\right) \tag{11}$$

with the Gaussian-like function $g_{\boldsymbol{\theta}}(x)$ defined in (8). Here the parameter $\theta_{e,1}$ is set to $\theta_{e,1} := 1$ such that $0 < g_{\boldsymbol{\theta}_e}\left(e_y^{\mathrm{P}}\right) \leq 1$ for all $e_y^{\mathrm{P}} \in \mathbb{R}$.

Since $g_{\boldsymbol{\theta}_\psi}\left(e_\psi^{\mathrm{P}}\right)$ and $g_{\boldsymbol{\theta}_v}\left(e_{v_x}^{\mathrm{P}}\right)$ are also bounded from above by $\theta_{\psi,1}$ and $\theta_{v,1}$ the whole term r_e tends to 0 for large position errors e_y^{P}. For small position errors e_y^{P} the value of $g_{\boldsymbol{\theta}_e}\left(e_y^{\mathrm{P}}\right)$ tends to 1 and the summands $g_{\boldsymbol{\theta}_\psi}\left(e_\psi^{\mathrm{P}}\right)$ and $g_{\boldsymbol{\theta}_v}\left(e_{v_x}^{\mathrm{P}}\right)$ determine the value of r_e. This means that the term r_e takes its maximum for zero tracking errors $0 = e_y^{\mathrm{P}} = e_\psi^{\mathrm{P}} = e_{v_x}^{\mathrm{P}}$ and tends to 0 if e_y^{P} is large. Additionally, the orientation error e_ψ^{P} and velocity error $e_{v_x}^{\mathrm{P}}$ dominate the value only if the position error e_y^{P} is small.

Designing the first term of the reward function (10), as proposed in equation (11), considers the position tracking as the primary control goal and velocity and orientation tracking as

secondary goals. This fulfils the requirements for PFC which are described above. The second term of (10) is chosen to penalize large control jittering:

$$r_{\Delta\delta}\left(e_y^{\mathrm{P}}, \Delta\delta^{\mathrm{f}}, \Delta\delta^{\mathrm{r}}\right) := g_{\boldsymbol{\theta}_{\Delta\delta}}\left(e_y^{\mathrm{P}}\right)\left(h_{\boldsymbol{\beta}_{\Delta\delta}}\left(\Delta\delta^{\mathrm{f}}\right) + h_{\boldsymbol{\beta}_{\Delta\delta}}\left(\Delta\delta^{\mathrm{r}}\right)\right). \tag{12}$$

The parameter $\theta_{\Delta\delta,1}$ is chosen as $\theta_{\Delta\delta,1} := 1$ such that the summands in the parenthesis of equation (12) are scaled by a factor between 0 and 1 depending on the position error e_y^{P}. As in the design of r_e this accounts for position tracking being the main control goal. The term $h_{\boldsymbol{\beta}_{\Delta\delta}}\left(\Delta\delta^i\right)$, with a dead zone-based function $h_{\boldsymbol{\beta}}$ according to (7), adds a negative value if the commanded steering angle δ_t^i exceeds the last clipped steering angle $\delta_{c,t-1}^i$ in magnitude by a margin $\Delta\delta_{\mathrm{lim}} =: \beta_{\Delta\delta,1}$, which guides the RL agent to favor smooth commanded steering angles.

For the variant with torque vectoring (RL-PFC TV) the reward function presented in (10) is augmented by an additional term penalizing the loss power $P_s^{\mathrm{W}_i}$ induced by the slip at each wheel:

$$\begin{aligned} &r_{\mathrm{RL-PFC\ TV}}\left(e_y^{\mathrm{P}}, e_\psi^{\mathrm{P}}, e_{v_x}^{\mathrm{P}}, \Delta\delta^{\mathrm{f}}, \Delta\delta^{\mathrm{r}}, P_s^{\mathrm{W}_1}, P_s^{\mathrm{W}_2}, P_s^{\mathrm{W}_3}, P_s^{\mathrm{W}_4}\right) \\ &\quad = r_{\mathrm{RL-PFC}}\left(e_y^{\mathrm{P}}, e_\psi^{\mathrm{P}}, e_{v_x}^{\mathrm{P}}, \Delta\delta^{\mathrm{f}}, \Delta\delta^{\mathrm{r}}\right) \underbrace{-d_1 \sum_{i=1}^{4}\left(\frac{P_s^{\mathrm{W}_i}}{d_2}\right)^2}_{r_{P_s}} \end{aligned} \tag{13}$$

with parameters $d_1, d_2 > 0$. The loss power can be calculated as follows:

$$P_s^{\mathrm{W}_i} = \boldsymbol{F}^{\mathrm{W}_i} \cdot \boldsymbol{v}_s^{\mathrm{W}_i} \text{ with } i \in \{1,2,3,4\} \tag{14}$$

with $\boldsymbol{v}_s^{\mathrm{W}_i} = \left[v_{s_x}^{\mathrm{W}_i}, v_{s_y}^{\mathrm{W}_i}\right]$ denoting the slip velocity vector and $\boldsymbol{F}^{\mathrm{W}_i}$ the contact force vector at wheel i (cf. Fig. 1) with $\boldsymbol{v}_s^{\mathrm{W}_i}$ and $\boldsymbol{F}^{\mathrm{W}_i}$ dependant on $\boldsymbol{a}$.

This augmentation guides the choice of the additional degrees of freedom in the RL-PFC TV compared to the RL-PFC. In contrast to all terms in $r_{\mathrm{RL-PFC}}$ (cf. equation (11) and (12)), the additional term in (13) is not multiplied by a term accounting for position tracking. This multiplier is dropped to guide the control allocation within the RL agent in an early stage of the training process, where position tracking is not accurate yet.

3.2.4 Observation Vector

The observation vector $\boldsymbol{s}_t$ which is fed back into the RL agent is composed of the errors defined in (3), the path curvature $\kappa_{\mathrm{P},t}$ as well as the clipped steering angles $\delta_{c,t}^{\mathrm{f}}$ and $\delta_{c,t}^{\mathrm{r}}$ at time step t. Since no memory is present in the policy the observation vector $\boldsymbol{s}_t$ is augmented by the values of the last time step $t-1$, yielding:

$$\boldsymbol{s}_t = \left[e^{\mathrm{P}}_{y,t}, e^{\mathrm{P}}_{v_x,t}, e^{\mathrm{P}}_{v_y,t}, e_{\psi,t}, \kappa_{\mathrm{P},t}, \delta^{\mathrm{f}}_{c,t}, \delta^{\mathrm{r}}_{c,t}, e^{\mathrm{P}}_{y,t-1}, e^{\mathrm{P}}_{v_x,t-1}, e^{\mathrm{P}}_{v_y,t-1}, e_{\psi,t-1}, \kappa_{\mathrm{P},t-1}, \delta^{\mathrm{f}}_{c,t-1}, \delta^{\mathrm{r}}_{c,t-1}\right] \quad (15)$$

Augmenting the observation vector with the values of the last time step brings rate information to the RL agent. It is also necessary to feed back the current clipped steering angles $\delta^{\mathrm{f}}_{c,t}$ and $\delta^{\mathrm{r}}_{c,t}$ to enable the RL agent to learn to drive smooth and avoid jittering in the commanded steering angles.

3.3 Implementation Details of the Training Framework

The PFC problem was implemented by using the interface from OpenAI Gym [27] and the SAC implementation from the stable-baselines [28]. As depicted in Fig. 6 the vehicle model and the time independent path interpolation were implemented in Modelica, exported as FMU by Dymola and included in the Python framework. Utilizing two different FMUs enables a highly modular training framework and new vehicle configurations can be easily adapted.

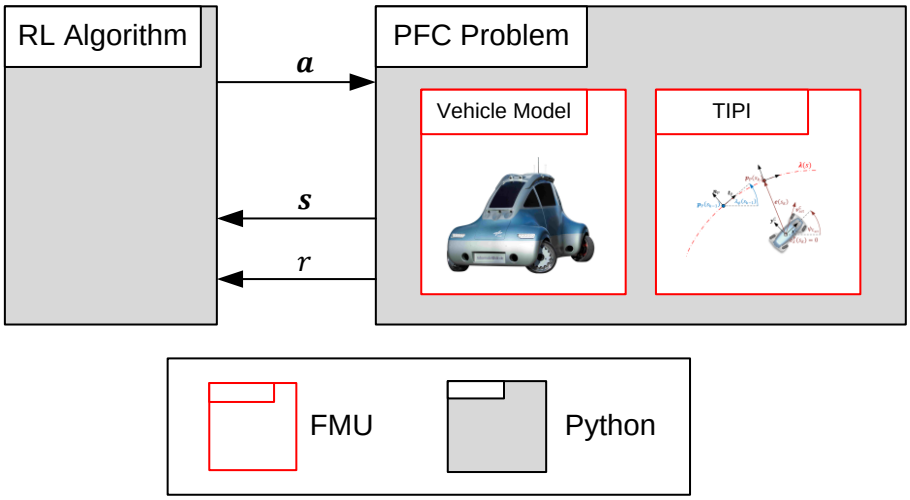

Fig. 6: RL-PFC implementation setup

The whole framework was implemented on a desktop PC with an Intel Xeon W-2135 CPU @ 3.70GHz and a Nvidia Quadro P600 [GP107GL] graphics card with openSUSE Leap 42.3 as operating system.

4 Simulative Assessment and Comparison

In this section the performance of the RL-PFC and RL-PFC TV controllers are evaluated and compared to a PFC reference controller.

The reference path following controller (later called ref PFC) used in this chapter is based on previous work in [6], [29]. To fulfil the control goal, a cascaded structure with three separate PD controllers stabilizing the position e^{P}_{y}, velocity $e^{\mathrm{P}}_{v_x}$ and orientation e_{ψ} tracking error in the outer control loop is used in the ref PFC. In the inner control loop an optimization based

control allocator was implemented in the reference controller. Details on the implementation are omitted here due to sake of brevity; the interested reader is referred to [6] for details.

4.1 Training of the RL-based Controller

For training the RL-based PFC a track on ADAC testing ground in Kempten is chosen (cf. Fig. 7a). The planned path on this test track is very demanding, includes tight and wide turns as well as straight lines and is, therefore, suited to expose the two RL-based PFC variants to challenging driving maneuvers during training.

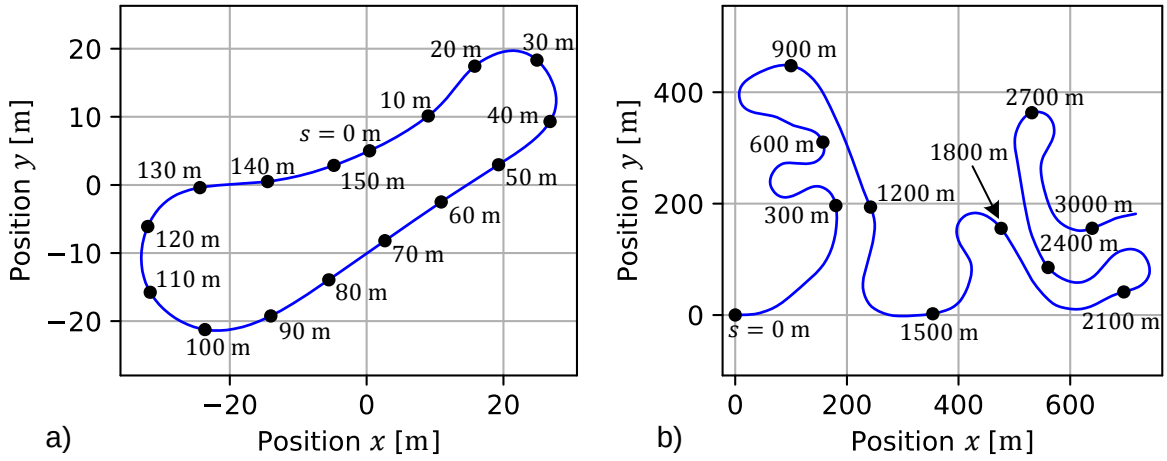

Fig. 7: a) Path used for training, b) Path used to show generalization

To avoid exploration of non-relevant regions and to guide the training, an episode is aborted if the vehicle exceeds certain position errors, orientation errors, etc. The abortion criteria have to be chosen carefully and are subject to a trade-off: On the one hand, early abortion guides the reinforcement learning and, therefore, a faster convergence during the training is given and thus shorter training times can be achieved. On the other hand, early abortion limits the room for exploring the observation space and, therefore, limits the robustness of the obtained controller. For our problem the following abortion criteria have been found to be convenient: An episode is aborted and a terminal reward of $r_T = -10$ is fed back if one of the following conditions is fulfilled: $\left|e_y^{\mathrm{P}}\right| > 2\ \mathrm{m}$, $\left|e_\psi\right| > 80°$, $\left|e_{v_x}^{\mathrm{P}}\right| > 2\ \frac{\mathrm{m}}{\mathrm{s}}$, $\left|e_{v_y}^{\mathrm{P}}\right| > 5\ \frac{\mathrm{m}}{\mathrm{s}}$, $|\boldsymbol{v}_{\mathrm{C}}| < 1\ \frac{\mathrm{m}}{\mathrm{s}}$. If an episode is aborted the vehicle is reinitialized at the abortion arc length. At the first episode and if the vehicle reaches the end of the path the vehicle is initialized at $s = 0\ \mathrm{m}$.

In reinforcement learning randomness in the initialization of the plant supports the exploration of the observation space. Therefore, the vehicle is initialized according to the desired motion demand $\boldsymbol{\lambda}(s)$ with some extra offset at the beginning of an episode. The offset is drawn from a uniform distribution within the following limits:

$$
\begin{aligned}
-1.4\ \mathrm{m} < e^{\mathrm{P}}_{y,\mathrm{start}} < 1.4\ \mathrm{m} \\
-24° < e_{\psi,\mathrm{start}} < 24° \\
-1\,\frac{\mathrm{m}}{\mathrm{s}} < e^{\mathrm{P}}_{v_x,\mathrm{start}} < 1\,\frac{\mathrm{m}}{\mathrm{s}}
\end{aligned}
\tag{16}
$$

It should be noted that choosing the limits too wide can result in situations where the abortion criteria will be violated no matter which input is chosen by the RL agent. Choosing the limits very restrictive will result in less exploration and, therefore, might yield a less robust controller.

In our use case the RL-PFC was trained with 300,000 and the RL-PFC TV variant with 400,000 time steps, which took 1h 15min and 1h 40min respectively. Fig. 8 depicts the episode return of the RL-PFC TV variant, which is the sum over all rewards during one episode, and the episode length, which is the number of time steps of one episode.

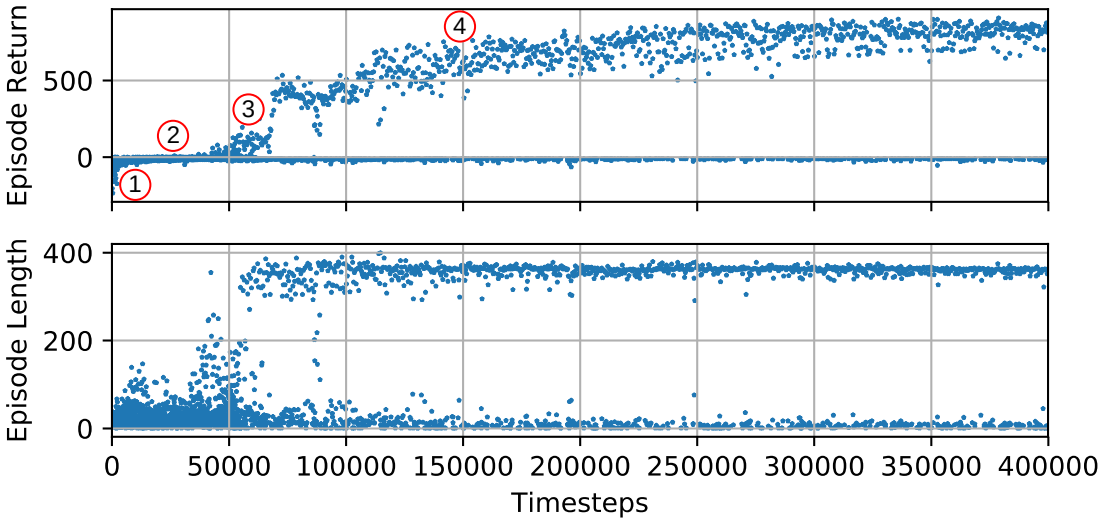

Fig. 8: Episode Length and Reward during training of RL-PFC TV

The training process can be divided into 4 different phases. In phase 1 the episode return is well below -10 and the episode lengths are short. The short episode length can be explained by early abortion due to violation of one of the abortion criteria defined above. This means that the agent is not able to solve the PFC task in the beginning. The low episode returns can be explained as follows: The terminal reward $r_T = -10$ can only be emitted once at the end of the episode and therefore all episode returns below -10 are caused by negative summands in the reward function (13). Since the only summands which can take negative values are $r_{\Delta\delta}$ and r_{P_s} they make up for the negative episode returns in the first phase caused by a jittery steering control and high slip losses.

By transitioning to phase 2 the RL agent learns to avoid jittery control and slip losses. Nevertheless, the agent still cannot master the task properly. This can be concluded based on the small episode returns as well as the short episode lengths during the second phase.
In phase 3 the algorithm more and more learns to solve the task, which can be seen in the longer episodes and higher returns.
During phase 4 the control policy is further improved to achieve higher rewards. In this phase the episode length and returns divide into two branches. One branch which stabilizes on a high level and one which is close to 0. The reason for the short episodes lies in cases where the vehicle is initialized in a challenging way and the exploratory behavior of the RL algorithm. As explained above this initialization is kept because more robust controllers could be obtained this way.

4.2 Comparison of the RL-PFC Variants with the Reference Controller on the Training Path

Table 1 shows the root mean square error (RMSE: $\sqrt{\frac{1}{s_{\max}}\int_0^{s_{\max}}(e(s))^2\,\mathrm{d}s}$) and the maximum absolute error ($\max_s|e(s)|$) of the two learning based controller variants with (RL-PFC TV) and without (RL-PFC) torque vectoring when executed on the path used for training (cf. Fig. 7a). To assess the performance of the learning-based controllers, the errors of the reference controller (ref PFC) are displayed as well. To evaluate the performance properly, the vehicle is initialized at $s = 0$ according to the motion demand $\boldsymbol{\lambda}(s = 0)$ with all errors equal to zero: $e_y^{\mathrm{P}} = 0,\ e_{v_x}^{\mathrm{P}} = 0,\ e_\psi = 0$.

Table 1: RMSE and maximum error on the training path with initialization according to the motion demand ($e_y^{\mathrm{P}} = 0,\ e_{v_x}^{\mathrm{P}} = 0,\ e_\psi = 0$)

		RL-PFC TV	RL-PFC	ref PFC
e_y^{P} [cm]	RMSE	**1.41**	1.60	2.39
	max	4.19	**3.55**	9.99
$e_{v_x}^{\mathrm{P}}$ [m/s]	RMSE	0.165	0.174	**0.0965**
	max	0.413	0.526	**0.383**
e_ψ [°]	RMSE	1.39	**0.679**	0.964
	max	3.39	**1.38**	2.66

From Table 1 it can be seen that both learning based controller variants are able to outperform the reference controller on the position error e_y^{P} on both error scales (RMSE and max). On the other hand, the velocity tracking performance of both RL variants is worse

than the reference controller (cf. Table 1). Both RL-PFC variants sacrifices velocity error in term of reduced position error. This originates in the selected reward function where we chose minimizing the position error to be the main priority. Nevertheless, the velocity tracking of learning based controllers is within the same number of magnitude as the ref PFC.

Even though position tracking was the main control goal the RL-PFC variant is able to outperform the ref PFC in minimizing the orientation tracking. On this path the RL-PFC TV is performing slightly worse in terms of orientation tracking than the ref PFC.

Overall the RL-PFC TV performs quiet similar compared to the RL-PFC variant which means that the RL-PFC TV is not able to exploit the additional degrees of freedom but is still able to allocate the degrees of freedom in the action space such that it can fulfil the PFC task.

4.3 Robustness Against an Initial Offset on the Training Path

To show PFC's ability to compensate for initial lateral position errors the vehicle was initialized with an initial lateral offset of $e_y^{\mathrm{P}} = -0.4\ \mathrm{m}$ on the path used for training (cf. Fig. 7a). All other states have been initialized according to the motion demand with $e_{v_x}^{\mathrm{P}} = 0$ and $e_\psi = 0$. In Fig. 9 the lateral position error e_y^{P}, longitudinal velocity error $e_{v_x}^{\mathrm{P}}$ and the orientation error e_ψ of all three controller variants are depicted.

It can be seen from Fig. 9 that all controller variants need approximately $20\ \mathrm{m}$ to eliminate the initial position tracking error.

Right after the recovery the vehicle has to follow the first sharp turn, which begins at approximately $s = 20\ \mathrm{m}$ and ends at approximately $s = 45\ \mathrm{m}$, leading into the long straight until approximately $s = 90\ \mathrm{m}$ (cf. Fig. 7a). It can be observed that both RL-PFC variants are able to track the path more accurately during the demanding first turn and during the first part of the straight. From the velocity profile it can be seen that the RL-PFC variants achieve the more accurate position tracking by adjusting the speed and therefore sacrifices velocity error in term of reduced position error. Since the position tracking is our main control goal we encouraged this kind of behavior by the design of the reward function described in section 3.2.3. The chosen reference controller is not able to deal with competing objectives since three separate PD controllers are used for position, velocity and orientation tracking.

It should be remarked that while driving on the straight between $s = 60\ \mathrm{m}$ and $s = 80\ \mathrm{m}$ a steady orientation error can be observed in all three applied controllers. Because of the over actuated vehicle architecture depicted in Fig. 1 it is possible to drive with an orientation error while maintaining the position error.

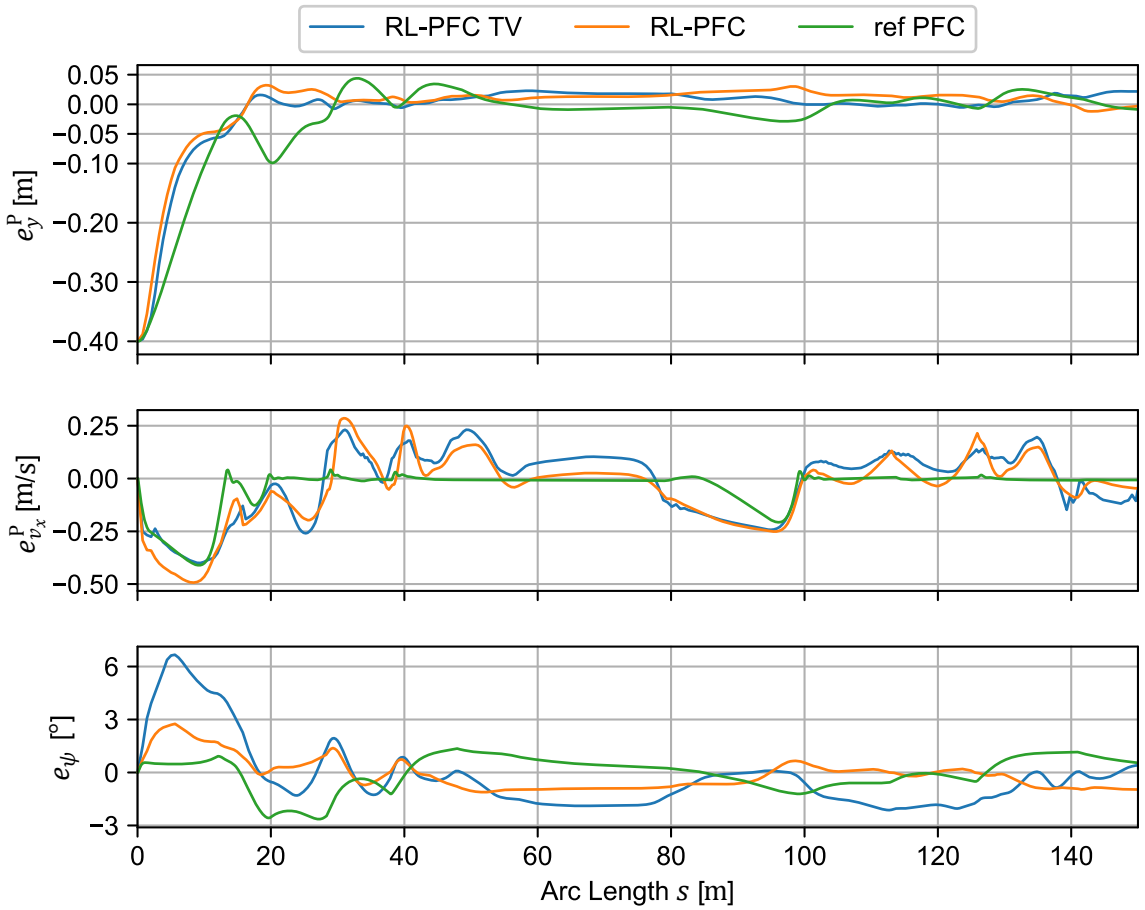

Fig. 9: Tracking errors on the training path with initial position error: $e_y^{\mathrm{P}} = -0.4\ \mathrm{m}$

The clipped commanded front- and rear axle steering angles δ_c^{f} and δ_c^{r} for driving on the training path are shown in Fig. 10. Since the reference PFC variant does not use the virtual control inputs δ^{f} and δ^{r} the average steering angle from the left and right hand side for the front and rear axles are used for comparison: $\delta_{\mathrm{ref}}^{\mathrm{f}} = \frac{\delta^{\mathrm{W}_1} + \delta^{\mathrm{W}_2}}{2}$ and $\delta_{\mathrm{ref}}^{\mathrm{r}} = \frac{\delta^{\mathrm{W}_3} + \delta^{\mathrm{W}_4}}{2}$. It can be observed that the overall control strategy of all three variants are quiet similar even though no a priori knowledge about the distribution of the steering angles had been fed into the RL-PFC design process.

Overall it can be concluded that in our example both RL-PFC could show its benefits especially in the most challenging driving situation which is the first sharp turn on our training course. Besides being able to handle the tradeoff between position-, velocity- and orientation tracking the RL-PFC variants do not rely on linear behavior and therefore can exploit the nonlinearity in this challenging driving situation.

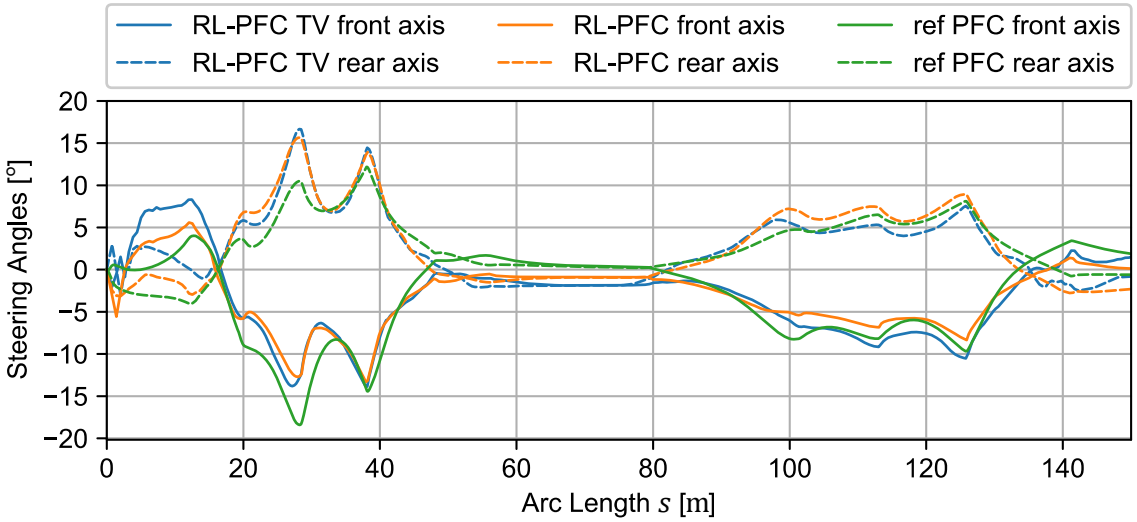

Fig. 10: Steering angles on the training path with initial position error: $e_y^{\mathrm{P}} = -0.4\ \mathrm{m}$

4.4 Evaluation of Generalization Capabilities of the RL-PFC Variants

In this section all control approaches are compared on a curvy road which is "unknown" to the learning based controller variants and is shown in Fig. 7b. Similar to the evaluation described in section 4.2 the vehicle is initialized with zero errors $(e_y^{\mathrm{P}} = 0,\ e_{v_x}^{\mathrm{P}} = 0, e_\psi = 0)$ according to the motion demand $\boldsymbol{\lambda}(s = 0)$. The RMSE and maximum errors are shown in Table 2.

Table 2: RMSE and maximum error on a curvy road with initialization according to the motion demand $(e_y^{\mathrm{P}} = 0,\ e_{v_x}^{\mathrm{P}} = 0, e_\psi = 0)$

Errors		RL-PFC TV	RL-PFC	ref PFC
e_y^{P} [cm]	RMSE	1.47	1.54	**0.172**
	max	2.59	3.51	**0.579**
$e_{v_x}^{\mathrm{P}}$ [m/s]	RMSE	0.130	0.142	**0.00652**
	max	0.311	0.314	**0.0140**
e_ψ [°]	RMSE	0.866	0.813	**0.121**
	max	2.29	1.39	**0.354**

Even though the curvy road is unknown to the RL-PFC variants, both RL-based controllers are able to follow the path with tracking error $\left|e_y^{\mathrm{P}}\right| < 3.6\ \mathrm{cm}$, velocity error $\left|e_{v_x}^{\mathrm{P}}\right| < 0.35\ \frac{\mathrm{m}}{\mathrm{s}}$ and orientation error $\left|e_\psi\right| < 2.3°$.

Still the reference controller is able to outperform the RL-based controllers on the curvy road. This can be explained by the fact that the path on the curvy road is a lot less challenging than the path used for training. Since the RL-PFC variants are exposed to a different motion profile in terms of curvature, velocity and orientation this region of the observation space is not well explored during training. Besides that, in section 4.3 it could be observed that RL has shown its advantage especially in challenging driving situations which are not present on the curvy road.

Even though only one path profile and a relative small number of training time steps was used for training the maximum position, velocity and orientation errors of the RL-PFC variants are within an acceptable range. To further improve generalization, the RL-PFC variants must be exposed to a greater variety of driving situations.

Moreover, it can be observed that the performance of the RL-PFC TV and the RL-PFC on all metrics displayed in Table 2 is quiet similar. This backs the observation from section 4.2 that the RL-PFC TV is not able to exploit the additional degrees of freedom in the action space but is still able to construct a control law such that an unknown path can be followed.

5 Conclusion and Outlook

In this work it has been shown that reinforcement learning-based path following controllers are able to outperform a model based reference controller [6] on the path used for training. Both proposed RL-based controllers could minimize an initial position error of $e_y^{\mathrm{P}} = -0.4\ \mathrm{m}$.

Despite training the RL-based path following controllers only on one path with one velocity profile they are able to follow an unknown path with a position, velocity and orientation error of $\left|e_y^{\mathrm{P}}\right| < 3.6\ \mathrm{cm}$, $\left|e_{v_x}^{\mathrm{P}}\right| < 0.35\ \frac{\mathrm{m}}{\mathrm{s}}$ and $\left|e_\psi\right| < 2.3°$, respectively.

Even though no human effort is necessary during training it should be noted that finding an appropriate reward function and hyper parametrization for the RL algorithm is not straight forward and includes trial and error. Another drawback is that for RL based controllers no formal stability guarantees can be derived yet.

The proposed RL-PFC TV variant with additional degrees of freedom in the action space could not exploit additional degrees of freedom but still allocated them in a way such that the control goal could be achieved.

Further research will address training strategies that enable a greater generalization and investigate robustness against uncertainties, and noise. Moreover, more effort will be spent on investigating possibilities to exploit over actuation.

6 References

[1] de Castro, R., Tanelli, M., Araujo, R., Savaresi, S.: Minimum-Time Path-Following for Highly Redundant Electric Vehicles. IEEE Transactions on Control Systems Technology 24 (2016) pp.487-501

[2] Falcone, P., Tseng, H., Borrelli, F., Asgari, J., Hrovat, D.: MPC-based yaw and lateral stabilisation via active front steering and braking. Vehicle System Dynamics 46 (2008) pp.611-628

[3] Lapierre, L., Soetanto, D., Pascoal, A.: Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties. International Journal of Robust and Nonlinear Control 16 (2006) pp.485-503

[4] Raffo, G., Gomes, G., Normey-Rico, J., Kelber, C., Becker, L.: A Predictive Controller for Autonomous Vehicle Path Tracking. IEEE Transactions on Intelligent Transportation Systems 10 (2009) pp.92-102

[5] Ritzer, P., Winter, C., Brembeck, J.: Experimental validation of geometric path following control with demand supervision on an over-actuated robotic vehicle. IEEE Intelligent Vehicles Symposium (IV) (2016) pp.539-545

[6] Brembeck, J.: Model Based Energy Management and State Estimation for the Robotic Electric Vehicle ROboMObil. Dissertation, Technische Universität München, München (2018)

[7] Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking Deep Reinforcement Learning for Continuous Control. Proceedings of the 33rd International Conference on Machine Learning (ICML) (2016) pp.1329-1338

[8] Koch, W., Mancuso, R., West, R., Bestavros, A.: Reinforcement Learning for UAV Attitude Control. ACM Transactions on Cyber-Physical Systems 3 (2019) pp.1-21

[9] Martinsen, A., Lekkas, A.: Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning. IFAC-PapersOnLine 51 (2018) pp.329-334 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS).

[10] Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., Shah, A.: Learning to Drive in a Day. arXiv eprint arXiv:1807.00412 (2018)

[11] Ritzer, P., Winter, C., Brembeck, J.: Advanced path following control of an overactuated robotic vehicle. IEEE Intelligent Vehicles Symposium (IV) (2015) pp.1120-1125

[12] Modelica Association: Functional Mockup Interface. (Accessed May 2019) Available at:

https://www.fmi-standard.org/

[13] Brembeck, J., Ho, L., Schaub, A., Satzger, C., Hirzinger, G.: ROMO - The robotic electric Vehicle. 22nd IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks (2011)

[14] Zimmer, D.: A Planar Mechanical Library for Teaching Modelica. Proceedings of the 9th International MODELICA Conference (2012) pp.681-690

[15] Zimmer, D., Otter, M.: Real-time models for wheels and tyres in an object-oriented modelling framework. Vehicle System Dynamics 48 (2010) pp.189-216

[16] Winter, C., Ritzer, P., Brembeck, J.: Experimental investigation of online path planning for electric vehicles. IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (2016) pp.1403-1409

[17] Brembeck, J.: Nonlinear Constrained Moving Horizon Estimation Applied to Vehicle Position Estimation. Sensors (2019) Accepted for Publication.

[18] Morin, P., Samson, C.: Motion Control of Wheeled Mobile Robots. In: Springer Handbook of Robotics. Springer Berlin Heidelberg (2008) pp.799-826

[19] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction 2nd edn. The MIT Press (2018)

[20] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. Proceedings of the 35th International Conference on Machine Learning (ICML) (2018) pp.1861-1870

[21] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. arXiv eprint arXiv:1707.06347v2 (2017)

[22] Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. International Conference on Learning Representations (ICLR) (2016) arXiv eprint arXiv:1509.02971.

[23] Ziebart, B.: Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, USA (2010)

[24] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft Actor-Critic Algorithms and Applications. arXiv eprint arXiv:1812.05905 (2019)

[25] Heißing, B., Ersoy, M., Gies, S.: Fahrwerkhandbuch: Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven 4th edn. Springer Nature (2013)

[26] Koch, W., Mancuso, R., Bestavros, A.: Neuroflight: Next Generation Flight Control Firmware. arXiv eprint arXiv:1901.06553 (2019)

[27] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym. (Accessed May 2019) Available at: https://github.com/openai/gym

[28] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y.: Stable Baselines. (Accessed May 2019) Available at: https://github.com/hill-a/stable-baselines

[29] Brembeck, J., Ritzer, P.: Energy optimal control of an over actuated Robotic Electric Vehicle using enhanced control allocation approaches. IEEE Intelligent Vehicles Symposium (IV) (2012) pp.322-327