Multibody System Dynamics
https://doi.org/10.1007/s11044-026-10144-x

RESEARCH

Path following and stabilization of a bicycle model using a reinforcement learning approach

Sebastian Weyrer[1] · Peter Manzl[1] · A.L. Schwab[2] · Johannes Gerstmayr[1]

Received: 28 July 2025 / Accepted: 15 January 2026

Abstract
Over the years, complex control approaches have been developed to control the motion of a bicycle. Reinforcement Learning (RL), a branch of machine learning, promises to be an automated approach for solving optimal control problems. By interacting with and observing an environment, a so-called agent is trained, ultimately leading to a learned controller. The present work introduces a pure RL approach to do path following with a virtual bicycle model while simultaneously stabilizing it laterally. The bicycle, modeled using the Whipple benchmark model and multibody system dynamics, has no stabilization aids. The observation of the environment consists of the minimal positional and velocity coordinates of the bicycle, as well as of information about the path ahead of the bicycle provided by moving preview points. Both path following and stabilization of the bicycle model are achieved exclusively by controlling the steering angle setpoint of the bicycle. Curriculum learning is applied as a state-of-the-art training strategy. Different settings for the RL approach are investigated and compared. The ability of the learned controllers to do path following and stabilization of the bicycle model traveling between 2 m/s and 7 m/s along complex paths including full circles, slalom maneuvers, and lane changes is demonstrated. Explanatory methods for machine learning are used to analyze the learned controller and identify connections to research in bicycle dynamics.

Keywords Reinforcement learning · Path following · Stabilization · Bicycle model · Curriculum learning · Machine learning

✉ J. Gerstmayr
johannes.gerstmayr@uibk.ac.at

S. Weyrer
sebastian.weyrer@uibk.ac.at

P. Manzl
peter.manzl@uibk.ac.at

A.L. Schwab
a.l.schwab@tudelft.nl

[1] Department of Mechatronics, University of Innsbruck, Technikerstraße 13, Innsbruck, 6020, Tyrol, Austria

[2] Department of Biomechanical Engineering, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, South Holland, The Netherlands

Published online: 29 January 2026

1 Introduction

Bicycle dynamics is a subject within dynamics studying the motion of bicycles under the influence of forces. Understanding bicycle dynamics is essential for developing controllers for bicycle models.

There is an increasing trend in multibody systems to apply machine learning methods, in the range between feed-forward networks for surrogate models [1] and large language models to model dynamic systems [2]. Also to address control problems, Reinforcement Learning (RL), a sub-area of machine learning, has been successfully applied to various multibody systems [3]. The idea of RL is that an agent learns to solve a task by observing the dynamic environment, receiving a reward signal, and returning actions to the dynamic environment without being supervised [4, 5]. The agent thus learns an observation-action mapping that is deployed as a learned controller. RL provides a general framework to address optimal control problems [6].

The question arises of whether RL could serve as a viable approach for controlling bicycles modeled as multibody systems. Answering this research question is of theoretical interest by showing the applicability of RL alongside other control approaches and holds practical potential for autonomous two wheeled systems like the one build in [7]. Towards answering the question, the state-of-the-art is considered from two perspectives in the following: bicycle control methods apart from RL and RL related approaches.

The control of the motion of a bicycle has attracted many researchers over the years [8]. Probably because of the speed-dependent lateral stability and the non-minimal phase behavior, that is, to stabilize a bicycle one has to steer into the direction of the undesired fall [9]. Most of the attention has been directed towards stabilizing the lateral dynamics of the bicycle and not so much on path following. A good overview on the dynamics and control of a bicycle is presented in [10]. After 2013, more work emerged on the control of the bicycle. The focus of the present work is on path following, which of course must include lateral stability control. A passive velocity field controller for path following has been introduced by Yin and Yamakita [11]. Their system is then passive with respect to external force perturbations. Turnwald et al. [12] introduce a passivity-based path following control for a bicycle. The steering and forward speed are controlled using a generalized canonical transformation on their port-Hamiltonian system of the bicycle. A classic multi-loop control structure is used by Shafiei and Emami [13] for path following of an unmanned bicycle. A classic LQR based controller for straight path following and cornering for a bicycle robot has been designed and built by Seekhao et al. [14], where the lateral dynamics of the bicycle is stabilized by an additional non-inverted pendulum. A model predictive controller with constraints on the lean, steer, heading as well as position of the bicycle is used by Persson et al. [15] for the design of a path following controller for a riderless bicycle. He et al. [16] present a learning based control framework for path following of a bicycle robot. For path following both steer and forward speed actuation are used. The lateral dynamics of the bicycle robot is stabilized by a controlled non-inverted pendulum. In the German-language doctoral thesis [7], a control law for the lateral stabilization of a slow-moving bicycle is derived and analyzed using a multibody simulation. A pedelec prototype is built to test the assistance system in the real world.

The idea of using an RL approach to control bicycle models is not new. An article from Randlov and Alstrøm [17] published in 1998 shows that RL can be used to keep a bicycle model upright and steer it to a target point that is 1000 m away from the initial position of the vehicle. The paper suggests using shaping, where two separate neural networks are used. The articles from Le and Chung [18] and Le et al. [19], published in the late years of 2010, show

that shaping is not necessary when further evolved RL algorithms are used, as a single agent learns to steer the bicycle model to the target point without it tipping over. The latter two works mainly adopt the bicycle model from [17]. In an approach that does not use RL per se, but shows that neural networks, in general, can be useful for controlling bicycles, Cook [20] manually devised a neural network controller with only two neurons as an example of a simple tracking controller. In the early years of 2020, Zhu et al. [21, 22] propose different RL approaches to do path following with a bicycle model even on curved pavements. The works do not consider lateral stabilization of the bicycle model as the therein used model has an integrated inertia wheel to enhance its stability. In a just published work, Huo et al. [23] introduce a hierarchical residual RL approach to do path following and stabilization of a bicycle model. Their approach uses the residual connections between different controllers. The RL based controllers are used as compensators for simplifications made for the design of the main controllers: For lateral stabilization, they are using a LQR based controller and for path following a Stanley [24] based controller. Huo et al. report a lack of convergence when using a purely RL based approach for path following and therefore do not pursue this approach further.

Despite various important contributions in the field of using RL with bicycle models, as of this writing, a research gap remains to the best of knowledge: the development of a purely RL based approach that can be used both to stabilize a bicycle model and to do path following along arbitrary paths.

The objective of the present work is to show that a purely RL based approach can be used to follow arbitrary paths with a benchmark bicycle model while simultaneously stabilizing it laterally across a range of forward velocities. The objective is pursued through the application of established RL methods in combination with state-of-the art training techniques and the systematic development of an RL environment. Findings in bicycle dynamics are incorporated in the analysis of the learned controller. The focus of the study is not to benchmark the RL approach against other control approaches.

In the present work, the bicycle model has no balance aids such as inertia wheels or non-inverted pendulums. The agent must learn to do path following and stabilization of the bicycle model exclusively by outputting set values for the steering angle of the bicycle. The bicycle in the present work is modeled using the so-called Whipple bicycle [25], which is regarded as a benchmark model to study bicycle dynamics [26]. The Whipple bicycle reflects the dynamic characteristics of the bicycle found on the road today, having a positive trail and being unstable when moving at low and high forward velocities and self-stable when moving at a specific speed range [8]. In intuitive control approaches, the speed-dependent lateral stability of the bicycle can be considered by switching between different control laws depending on the bicycle's forward velocity [27]. This adds an intriguing aspect to the present work, since the agent must implicitly learn the required regime-dependency to stabilize the Whipple bicycle. Thereby, the agent must learn to cope with different forward velocities, underlining the practical significance of the present work by addressing a system where the velocity of the bicycle is set by the cyclist while path following and stabilization of the bicycle is done by automated steering movements. By incorporating preview points that add information about the path ahead to the observation, the fully data-driven approach is expected to enable proactive behavior even in challenging path scenarios. Finally, by using explanatory methods for machine learning, the present work aims to verify that the learned controller shows the basic mechanisms of controlling a bicycle, closing the circle to bicycle dynamics.

An additional aspect of the present work is that the environment is virtual, with the bicycle implemented as a multibody model. Using a virtual environment is accompanied by

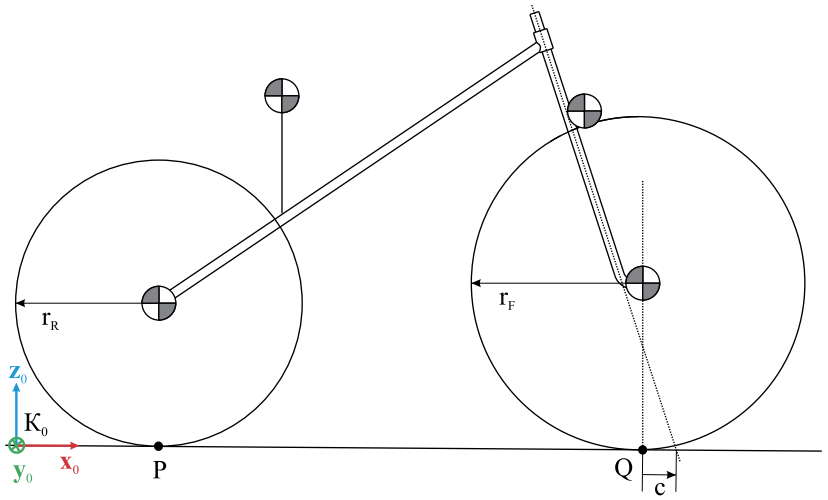

Fig. 1 Whipple bicycle shown in the reference configuration where it stands upright without the handlebar being turned. The position of the Center of Mass (COM) of the rear wheel, the rear body, the handlebar, and the front wheel is marked, as well as the two ground contact points P and Q, the two wheel radii r_R and r_F, and the trail c of the bicycle. The global frame, denoted as 0-frame, is shown

simplifications resulting from the bicycle model and its surrounding. It is assumed that the bicycle model drives on a perfectly leveled surface.

The simplifications and preparatory work associated with the bicycle model to apply an RL approach mark the beginning of this article. Afterwards, the applied RL framework is described, followed by an outline of the learning process with the application of curriculum learning as a training strategy. The subsequent section analyzes the learning process and evaluates how accurately the learned controllers can do path following and stabilization of the bicycle model at different forward velocities and along different types of paths. Explanatory methods for machine learning are used to analyze a learned controller. Finally, the presented results are discussed. To complement the work, videos of the virtual bicycle model are provided as supplementary material.

2 Bicycle model and associated preparatory work

The bicycle model used in the present work is the Whipple bicycle [25]. The Whipple bicycle is often used to study bicycle dynamics [26] and the linearized equations of motion for the Whipple bicycle are presented in a benchmark paper [8]. In this section, the bicycle model itself and the preparatory work that is required to use the bicycle model with the RL approach are described.

2.1 Characteristics

The Whipple bicycle, shown in Fig. 1, consists of four rigid bodies, namely the Rear wheel (R), the Front wheel (F), the rear Body (B) being the bicycle frame and the rider, and the Handlebar (H) that includes the fork. The bodies are interconnected by three frictionless revolute joints, one at the bicycle head tube and two at the wheel hubs. The Whipple bicycle

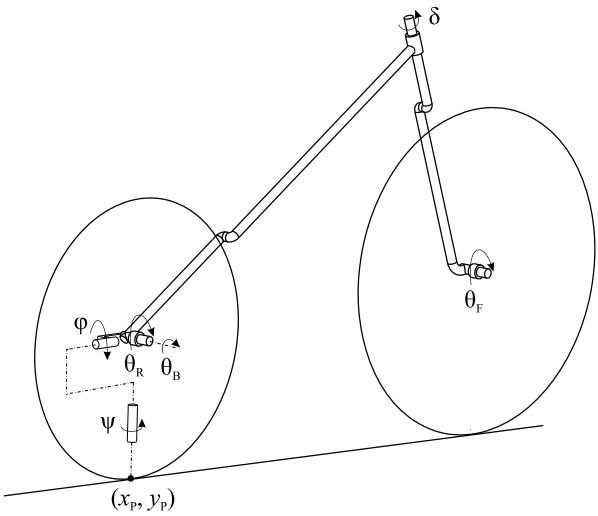

Fig. 2 Whipple bicycle drawn with its minimal coordinates on positional base and the pitch angle θ_B. The upright cylinder marked with Ψ represents the yaw angle, the mounting attached to the rear dropout is used to illustrate the roll angle φ that is independent of the pitch angle θ_B of the rear body. Note that the pitch angle θ_B is not a minimal coordinate of the bicycle, but is needed later for the coordinates mappings

has a tilted steering axis and a fork offset. Due to the fork offset of the bicycle, the front wheel hub is not located on the steering axis, but at a constant distance to it. The Whipple bicycle has a positive trail c. The rider body is assumed to be rigidly connected to the frame of the bicycle. The wheels are modeled using geometrically ideal thin discs, each having one contact point with the flat level ground. The contact points of the wheels with the ground are called P and Q. The geometrical and mechanical parameters of the bicycle model are taken from [8]. The bicycle model is subjected to gravitational acceleration. The bicycle model is self-stable when traveling at a forward velocity between 4.3 ms^{-1} and 6 ms^{-1} [8], as initial perturbations of the bicycle model subside over time if moving in this range for the forward velocity.

2.2 Minimal coordinates

To use RL, an observation must be defined which describes the environment in such a way that the agent can learn selecting a suitable action [4]. In the present work, the observation includes the minimal coordinates of the bicycle model. For the degree of freedom on positional base, $f_L = 6n - b_L = 24 - 17 = 7$ follows, where n is the number of rigid bodies of the system and b_L the number of geometric constraints in the system [8]. The number of geometric constraints is the sum of the number of constraints resulting from the revolute joints plus two, as the wheels cannot penetrate the ground. In Fig. 2, the minimal coordinates on positional base of the system are illustrated. Since the bicycle has non-holonomic constraints, the degree of freedom on velocity base is further reduced to $f = f_L - b^* = 7 - 4 = 3$, where b^* is the number of non-holonomic constraints [8]. Per wheel, two non-holonomic constraints are added since neither side-slipping nor free spinning of the wheels is possible in the used bicycle model. In Table 1, the minimal coordinates on positional and velocity base are given. The vector $\mathbf{q} \in \mathbb{R}^{10}$ of the minimal coordinates reads

$$\mathbf{q} = \begin{bmatrix} x_P & y_P & \Psi & \varphi & \delta & \theta_R & \theta_F & \dot{\varphi} & \dot{\delta} & \dot{\theta}_F \end{bmatrix}^T . \tag{1}$$

Table 1 Minimal coordinates of the Whipple bicycle that are part of the observation describing the environment. The T_1-frame used is defined in Appendix A

Notation	Description
x_P	x-coordinate of the global position of the rear wheel contact point
y_P	y-coordinate of the global position of the rear wheel contact point
Ψ	yaw angle between the x-axes of the 0-frame and the T_1-frame around the z-axis of the 0-frame
φ	roll angle around the x-axis of the T_1-frame, where $\varphi = 0$ means that the z-axes of the T_1-frame and the 0-frame are parallel
δ	steering angle
θ_R	rear wheel rotation angle
θ_F	front wheel rotation angle
$\dot{\varphi}$	roll angular velocity
$\dot{\delta}$	steering angular velocity
$\dot{\theta}_F$	front wheel rotation angular velocity

Table 2 Description of the coordinates (coord.) mappings that must be established for using the multibody model of the bicycle in the RL framework

Redundant → Minimal	Minimal → Redundant
compute the minimal coord. (on positional and velocity base) from the current configuration of the bicycle since the minimal coord. are part of the observation describing the environment; without this transformation the observation of the environment could not be computed	compute the redundant coord. (on positional and velocity base) when the environment is reset since the environment is reset by randomly initializing the minimal coord. of the bicycle model; without this transformation the simulation could not restart after the environment is reset

2.3 Coordinates mappings

For the multibody model, a redundant formulation is used, representing the configuration of each rigid body with a translation and rotation. The redundant formulation avoids the need to derive the equations of motion specifically for the bicycle model, as it relies on tested components implemented within a multibody simulation framework. Furthermore, the redundant formulation facilitates extensions of the bicycle model. The redundant formulation could naturally accommodate the loss of wheel contact on uneven pavement, which would result in a variable degree of freedom throughout a simulated ride. Further examples for a future extension are the use of flexible bodies, each of which could be modeled using the floating frame of reference formulation [28], or the inclusion of joint clearance, which would impose unilateral constraints on the system [29]. By using a redundant formulation, a mapping of the redundant coordinates to the minimal coordinates and vice versa must be established before setting up the RL framework. In Table 2, the two mappings are described, as well as why the two mappings are required.

In Appendix A, the mapping of the redundant to the minimal coordinates and the mapping of the minimal to the redundant coordinates are shown. The mappings, which are also useful for work outside the RL context, include both the positional and velocity base.

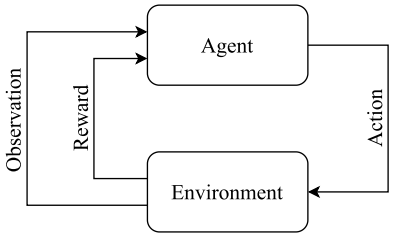

Fig. 3 Scheme of the RL framework. An agent can choose an action based on the observation of a dynamic environment. A numerical reward signal is passed pack to the agent

2.4 Model of the steering drive

The bicycle model is controlled by the set value δ_{set} for the steering angle of the bicycle model. The set steering angle is converted into a steering torque τ by a PD controller. The steering torque τ is imprinted between the rear body and the fork of the bicycle model and acts around the steering axis, modeling a drive sitting in the bicycle head. The steering torque τ reads

$$\tau = P(\delta - \delta_{\text{set}}) + D(\dot{\delta} - \dot{\delta}_{\text{set}}), \tag{2}$$

where δ is the steering angle of the bicycle model and $\dot{\delta}$ is the steering angular velocity. For the set value of the steering angular velocity, $\dot{\delta}_{\text{set}} = 0$ applies, as the steering angular velocity $\dot{\delta}$ is not controlled. The two parameters of the PD controller are set $P = 9$ Nm and $D = 1.6$ Nms. In Appendix B, it is described how the values for P and D are found.

3 Reinforcement learning framework

In an RL framework, the agent can choose an action based on an observation describing the environment.[1] The action chosen by the agent changes the dynamic environment. The new observation and a reward are passed back to the agent, closing the agent-environment loop, see Fig. 3 [4]. Based on the new observation, the agent again selects an action. The reward is a numerical signal. The agent tries to maximize the cumulative reward. Learning means that the data obtained when passing through the agent-environment loop is used to adapt the behavior, i.e. the policy, of the agent. The method used to adapt the behavior is defined by the specific RL algorithm used. In the present work, mainly the off-policy Soft Actor Critic (SAC) algorithm is used, first introduced in the work [31]. The SAC algorithm is off-policy, so it uses a replay buffer, where the data obtained when running through the RL framework is saved. After every single run through the RL framework, a batch sampled from the replay buffer is used to adapt the behavior of the agent [31]. If the agent is implemented using neural networks, the concept of RL is extended to Deep Reinforcement Learning (DRL). Adapting the behavior of the agent in the context of DRL means that the weights and biases in the neural networks are adapted [5]. Within typical implementations, an SAC agent consists of five neural networks [32]. In the present work, the terms RL and DRL are used synonymously, as all agents are implemented using neural networks. After a

[1] Note that in the RL literature, the terms *observation* and *state* are often used interchangeably. Strictly speaking, the state is a complete non-redundant description of a system, while the observation may be incomplete or includes redundant information [30]. Since the agent in this work receives the state of the bicycle, i.e. the minimal coordinates on positional and velocity base, refer to Sect. 2.2, but rather than the complete information about the path only a preview of the path, the agent effectively receives an observation.

specified number of learning steps, or once a designated performance criterion is met during training, the behavior of the agent is no longer adapted. The observation–action mapping of the agent is deployed as a controller. In the case of the SAC algorithm, the deployed controller consists of the actor. The actor is one of the five neural networks that constitute an SAC agent. The actor represents an observation-conditioned probability distribution over the actions [31] by outputting means and standard deviations for each action dimension. To act as a deterministic learned controller, the mean action is selected for each observation, effectively functioning as a feedback controller [30]. The reader should note the following distinction used throughout this article:

- **Agent**: The agent learns in a data-driven manner within the agent–environment loop and uses neural networks internally to adapt its behavior.
- **Learned Controller**: The learned controller is the observation–action mapping acquired by the agent, i.e., a feedback controller. It consists of components of the agent (e.g., the learned actor) but remains fixed and is not adapted during operation.

The framework set up for the present work is Python (version 3.11.5) based. The multibody simulation code Exudyn[2] is used for the multibody simulation of the bicycle model. Exudyn is based on C++ and available as Python package [33]. Exudyn provides an interface class that connects the multibody system to OpenAIGym[3] [34]. OpenAIGym in turn provides a standardized port to already existing implementations of RL algorithms. The RL algorithms used in the present work are part of Stable-Baselines3[4] [32], which provides implementations of seven different RL algorithms.

In the following, four main parts of the RL framework are described, namely the environment, the observation, the reward, and the action. At certain points in the following descriptions, several possible settings for the RL framework are given. The different settings for the RL framework are compared to each other later when presenting the results.

3.1 Environment

The environment consists of the simulated bicycle model and the path the bicycle should follow. As outlined in Sect. 2.3, the multibody model of the bicycle is implemented using a redundant formulation. The differential algebraic equations resulting from the redundant formulation and assembled by Exudyn are solved using the generalized-α solver implemented in Exudyn. The step size of the solver is set 0.005 s. The RL algorithm interacts every $h = 0.05$ s with the environment, with $1/h$ being the controller frequency. Thus, 10 simulation steps are made until a new action is applied to the multibody system. The update time h is set higher than the step size of the solver, so that the dynamics of the environment can be learned in fewer interactions between the agent and the environment. The visualization of the bicycle model is shown in Fig. 4, left.

The paths the bicycle model should follow are built using three different element types, namely linear elements, polynomial elements, and circular elements, see Fig. 4, right. The linear element is defined by its length L. The circular element is defined using an angle Φ and a radius R. The polynomial element is defined by its length L_P and width W_P. The polynomial elements are all of degree five. In the present work, two families of paths are used, see Fig. 5. For the learning of the agents, paths are initialized randomly in a wide

[2] https://github.com/jgerstmayr/EXUDYN (used version: 1.8.0).

[3] https://github.com/Farama-Foundation/Gymnasium (used version: 0.21.0).

[4] https://github.com/DLR-RM/stable-baselines3 (used version: 1.7.0).

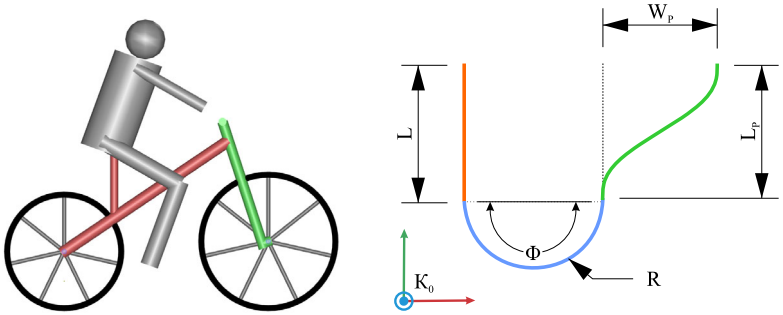

Fig. 4 (Left) The visualization of the virtual bicycle model is shown. (Right) Examples of the three element types with which the paths in the present work are constructed are drawn with their corresponding geometric properties

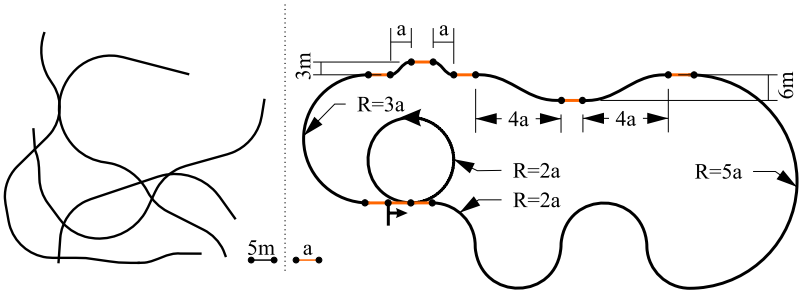

Fig. 5 (Left) Four examples of randomly generated paths which are used in the learning process are shown. Short parts of the paths are drawn each to prevent clustering in the figure. (Right) The benchmark path that is used to evaluate the performance of the learned controllers is shown. For the benchmark path, $a = 5$ m applies

range using the reset procedure, so that the agents not only learn to follow a specific path, but to follow arbitrary paths. For the performance evaluation of the learned controllers, additionally, a benchmark path is introduced, including a full circle, a slalom, curves, two lane changes, and straight elements.

3.2 Observation

The observation in the present work contains a potentially redundant formulation of the minimal coordinates of the bicycle model and preview information describing the path the bicycle should follow. Both parts are described now.

In the preparatory work associated with the bicycle model, its minimal coordinates $\mathbf{q}$ are defined, see Sect. 2.2 and Equation (1). One part of the observation vector $\mathbf{o}$ are the minimal coordinates, either in the form of $\mathbf{q}$ or in the form of an alternative redundant formulation $\mathbf{q}' \in \mathbb{R}^{11}$. In the alternative formulation, the yaw angle Ψ is represented using a unity vector with the two components

$$x_\Psi = \cos\Psi \qquad \text{and} \qquad y_\Psi = \sin\Psi\,. \tag{3}$$

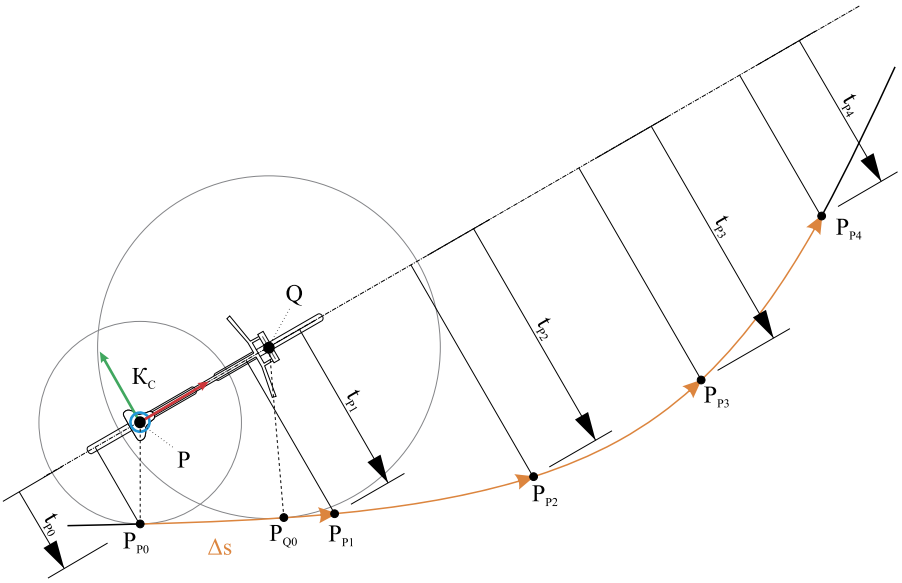

Fig. 6 Scheme of the procedure to get preview information of the path. The preview points required for the preview information are obtained by moving along the path by the preview distance Δs, starting with the point P_{P0}. The point P_{P0} is the point on the path nearest to the rear wheel contact point P. The point P_{Q0} is the point on the path nearest to the front wheel contact point Q and can be used later for the reward computation

Consequently, the alternative redundant formulation for $\mathbf{q}$ reads

$$\mathbf{q}' = \begin{bmatrix} x_\mathrm{P} & y_\mathrm{P} & x_\Psi & y_\Psi & \varphi & \delta & \theta_\mathrm{R} & \theta_\mathrm{F} & \dot{\varphi} & \dot{\delta} & \dot{\theta}_\mathrm{F} \end{bmatrix}^\mathrm{T} . \tag{4}$$

The preview information of the path is incorporated in the observation $\mathbf{o}$ using the vector $\mathbf{t}_\mathrm{P} \in \mathbb{R}^{(n_\mathrm{prev}+1)}$, where $n_\mathrm{prev} = 4$ is the number of preview points used to compute the vector $\mathbf{t}_\mathrm{P}$. The vector $\mathbf{t}_\mathrm{P}$ represents the course of the path ahead of the bicycle model given in the C-frame. The C-frame is defined in Equation (A.3). To obtain the vector $\mathbf{t}_\mathrm{P}$, it is started with the point P_{P0} illustrated in Fig. 6. The vector pointing from the rear wheel contact point to P_{P0} is projected on the y-axis of the C-frame. The length of the projection is the first entry of the vector $\mathbf{t}_\mathrm{P}$, denoted as t_{P0}. The point P_{P0} is moved along the path by the preview distance Δs. This gives the first preview point, called P_{P1}. Again, the vector that is pointing from P to P_{P1} is projected on the y-axis of the C-frame, which gives the second entry t_{P1} of the vector $\mathbf{t}_\mathrm{P}$. Repeating this procedure, the vector $\mathbf{t}_\mathrm{P}$ finally reads

$$\mathbf{t}_\mathrm{P} = \begin{bmatrix} t_{P0} & t_{P1} & t_{P2} & t_{P3} & t_{P4} \end{bmatrix}^\mathrm{T} . \tag{5}$$

Subsuming, the observation vector $\mathbf{o}$ follows as

$$\mathbf{o} = \begin{bmatrix} \mathbf{q} \\ \mathbf{t}_\mathrm{P} \end{bmatrix} \quad \text{or} \quad \mathbf{o} = \begin{bmatrix} \mathbf{q}' \\ \mathbf{t}_\mathrm{P} \end{bmatrix} . \tag{6}$$

The two formulations for the observation vector $\mathbf{o}$ are compared to each other in the results section, as are two ways to set the preview distance Δs.

3.3 Reward

To represent the two components of the here investigated task of doing path following with a bicycle model and stabilizing it laterally, the reward $\rho \in [0, 1]$ is the linear combination of the two parts ρ_y and ρ_φ.

$$\rho = \chi_1 \cdot \rho_\mathrm{y} + \chi_2 \cdot \rho_\varphi \tag{7}$$

For the two reward weights $\chi_1 + \chi_2 = 1$ applies.

The first part $\rho_\mathrm{y} \in [0, 1]$ rewards the agent when the bicycle is close to the path. A distance referred to as t_0 (not to be confused with $t_{\mathrm{P}0}$ in the preview vector, see Sect. 3.1) is used. The distance t_0 can either be the Euclidean distance $d(\mathrm{P}, \mathrm{P}_{\mathrm{P}0})$ between the rear wheel contact point P and $\mathrm{P}_{\mathrm{P}0}$ or the Euclidean distance $d(\mathrm{Q}, \mathrm{P}_{\mathrm{Q}0})$ between the front wheel contact point Q and $\mathrm{P}_{\mathrm{Q}0}$. The points $\mathrm{P}_{\mathrm{P}0}$ and $\mathrm{P}_{\mathrm{Q}0}$ are shown in Fig. 6. A relationship must be introduced to compute the part ρ_y with the distance t_0. Either a linear relationship reading

$$\rho_\mathrm{y} = \rho_\mathrm{y}^\mathrm{lin} = \frac{\eta_\mathrm{y} - t_0}{\eta_\mathrm{y}} \tag{8}$$

or a relationship forming the normalized Gaussian bell curve, reading

$$\rho_\mathrm{y} = \rho_\mathrm{y}^\mathrm{exp} = \exp\left(-0.5 t_0^2\right) \tag{9}$$

can be used. The distance threshold η_y is set to 3.5 m. As explained later, the environment is reset if the bicycle model drives further away than η_y from the path. Consequently, when the distance between the bicycle model and the path becomes maximum, the part $\rho_\mathrm{y}^\mathrm{lin}$ of the reward equals zero.

The part $\rho_\varphi \in [0, 1]$ is computed using the roll angle φ.

$$\rho_\varphi = \frac{\eta_\varphi - |\varphi|}{\eta_\varphi} \tag{10}$$

By computing the part ρ_φ this way, the agent is penalized if the bicycle leans. The roll angle threshold η_φ is set to 45°. If the magnitude of the roll angle φ exceeds η_φ, the environment is reset. Consequently, when the magnitude of the roll angle φ becomes maximum, the part ρ_φ of the reward equals zero. Although this reward part is conflicting with training the agent to follow paths including curves, since with a curve the roll angle φ must be non-zero, it will be investigated later whether and to what extent this interferes with the learning progress.

The two described options to obtain the distance t_0 will be compared to each other later, as well as using Equation (8) or Equation (9) to compute the part ρ_y. Furthermore, the influence of the reward weights χ_1 and χ_2 is analyzed.

3.4 Action

The bicycle model is controlled by taking steering actions. The control input for the bicycle model is the set value $\delta_\mathrm{set} \in [-70°, 70°]$ for the steering angle, as described in Sect. 2.4. The interval given for δ_set is substantiated in Appendix B.

4 Learning process with curriculum learning

The learning process is the process that generates data with the multibody simulation environment and uses this data to optimize the function approximators of the agent. The learning processes in the present work consist of $n_{\mathrm{tot}} = 4 \cdot 10^6$ learning steps, since, as shown later, the validation results no longer change significantly at this level of training. One learning step is one run through the RL framework. This means that one learning process uses data generated in $n_{\mathrm{tot}} \cdot h = 55.5$ hours of simulated bicycle rides, with h being the time a selected action is applied, see Sect. 3.1. The learning process can be divided into episodes, whereby the end of an episode can trigger certain procedures in the learning process, as described later. An episode ends when at least one of the conditions listed below is met:

- The episode consists of $n_{\mathrm{max}} = 1200$ learning steps, so that one episode represents a bicycle ride with a maximum duration of $n_{\mathrm{max}} \cdot h = 60$ s. Note that when this condition is met, the environment is not necessarily reset before the start of the next episode, see next subsection.
- At least one of the two thresholds η_{y} and η_{φ} is exceeded and the environment is reset. The two thresholds η_{y} and η_{φ} are defined in Sect. 3.3.

As long as the learning process does not contain n_{tot} learning steps, a new episode starts when the previous episode ended.

In addition to running through the RL framework, the learning process must reset the environment when certain conditions are met. Furthermore, validations are performed during the learning process to monitor the progress of learning. During the learning process, a training strategy called Curriculum Learning (CL) is applied. The reset procedure, the validation procedure, and the CL are explained now in more detail.

4.1 Resetting the environment

Primarily, the environment of the RL framework must be reset if the control task is failed and should be solved again. The reset procedure is therefore called when the roll angle φ of the bicycle model exceeds the roll angle threshold η_{φ} or/and when the distance between the rear wheel contact point P and the point P_{P0} exceeds the distance threshold η_{y}. In other words, reset is called when the bicycle has fallen over or/and driven far away from the path. If two consecutive episodes contain the maximum number of learning steps n_{max} and no reset was called between the episodes, the environment is also reset. Resetting the environment at this condition ensures that the learning agent that already succeeds in solving the control task without exceeding the thresholds η_{y} and η_{φ} still finds a variety of initial conditions to learn from. The reset procedure initializes the configuration of the bicycle model as well as the path the bicycle model should follow.

The configuration of the bicycle model is initialized by randomly setting the minimal coordinates of the bicycle model in the intervals given in the upper part of Table 3. The roll angle φ, the steering angle δ, and their angular velocities are not set to zero when initializing the bicycle model. By initially perturbing the bicycle model, the agent should learn how to stabilize the bicycle by just observing the beginnings of every simulated bicycle ride. The forward velocity of the bicycle model is set randomly between v_{min} and v_{max}. Note that the bicycle cannot accelerate or decelerate in the simulated bicycle rides, as described in Sect. 1. Since the forward velocity is not a minimal coordinate of the bicycle model, it is converted into the front wheel rotation angular velocity $\dot{\theta}_{\mathrm{F}}$ using

$$\dot{\theta}_{\mathrm{F}} = \frac{v}{r_{\mathrm{F}}}, \tag{11}$$

Table 3 Intervals in which the parameters of the environment are initialized when reset is called, where for each parameter x an interval $a \leq x \leq b$ is given. The table indicates which initialization intervals are used to apply CL, see also Table 4. For those parameters, no concrete interval can be given

Parameter	Initialization interval		CL
	a	b	
bicycle model			
coordinates x_P and y_P of the contact point P	-10 m	10 m	no
yaw angle Ψ	$-\pi$ rad	π rad	no
roll angle φ	-0.01 rad	0.01 rad	no
steering angle δ	-0.01 rad	0.01 rad	no
wheel rotation angles θ_R and θ_F	$-\pi$ rad	π rad	no
roll angular velocity $\dot{\varphi}$	-0.05 rads^{-1}	0.05 rads^{-1}	no
steering angular velocity $\dot{\delta}$	-0.01 rads^{-1}	0.01 rads^{-1}	no
front wheel rotation angular velocity $\dot{\theta}_F$	set with the forward velocity v		-
forward velocity v	v_{min}	v_{max}	yes
path			
length of linear elements L	5 m	15 m	no
angle of circular elements Φ	$\pi/4$ rad	$\pi/2$ rad	no
radius of circular elements R	R_{min}	R_{max}	yes
length of polynomial elements L_P	$L_{P,min}$	$L_{P,max}$	yes
width of polynomial segments W_P	$W_{P,min}$	$W_{P,max}$	yes

with r_F being the radius of the front wheel. The parameters not necessary for solving the control task, i.e. the contact point $P = (x_P, y_P)$, the yaw angle Ψ, and the wheel rotation angles θ_R and θ_F, are initialized in wide intervals so that the agent learns their irrelevance.

The randomly initialized path the bicycle model should follow lies in the plane spanned by the x- and the y-axes of the 0-frame. The beginning of the path is placed where the contact point P of the rear wheel of the bicycle is initialized. The beginning of the path is rotated so that the angle of the tangent of the path at the point P is inside the interval $[\Psi - 5°, \Psi + 5°]$, where Ψ is the yaw angle the bicycle is initialized with. The path is initialized by appending a random sequence of the three element types the path can consist of, see Sect. 3.1. Not only the sequence of element types, but also the geometric properties of each individual element of the path are initialized randomly in a given interval. The intervals for the geometric properties of the elements are given in the lower part of Table 3. The sign of the curvature of the circular elements is selected at random so that with a probability of 50% the element is positively or negatively curved. The same is done with the width $|W_P|$ of the polynomial elements to specify the direction of the transfer. It is specified that the initialized path does not start with a circular element so that after resetting, where the bicycle model may be perturbed, the task does not become unachievable. Additionally, it is defined that 40% of the elements in the initialized path should be circular elements, 40% should be polynomial elements, and 20% should be straight elements. This is specified so that the path does not contain long straight sections with which a behavior to follow the straight elements but not the other two types of elements could be exploited. The family of paths assembled in the way of the present work cannot be followed by the bicycle model exactly, because the dynamic characteristics of the bicycle impose limitations in its maneuverability [35]. In particular, step-like changes in the curvature of the path result in a path error between the

bicycle model and the path, alike a bicycle traveling on real roads. To start the turn of a bicycle, the handlebar must be turned in the opposite direction, known as counter-steering, so that the bicycle leaves the desired path [8, 35]. This dynamic property of the bicycle model must be learned by the agents, trying to follow the path as closely as possible.

4.2 Validation during the learning process

The validation procedure is to monitor the success of the learning process. In the learning process, a validation is always carried out if both of the following conditions are fulfilled: An episode reaches $n_{\max}$ learning steps and the last validation was 4000 or more learning steps ago. The first condition is imposed because it is assumed that, upon reaching the maximum number of steps in an episode, the task can already be partially solved and it makes sense to validate the performance. The latter condition is set to save computing time, as an already well-trained agent is not validated after each episode. A validation consists of 10 simulated bicycle rides of a duration of 30 s. To do path following and stabilization of the bicycle model in these simulated bicycle rides, the actor of the agent in its current state of training is used to get a learned controller, deterministically outputting set values for the steering angle of the bicycle based on received observations. Before each of the simulated bicycle rides in the validation, the environment is reset using the procedure and initialization intervals described in Sect. 4.1. Each simulated bicycle ride is then scored using a normalized error $e_i \in [0, 1]$.

$$e_i = \begin{cases} 1 & \text{if } \eta_\mathrm{y} \text{ or } \eta_\varphi \text{ is exceeded} \\ \frac{\bar{t}_{0,80\%}}{\eta_\mathrm{y}} & \text{else} \end{cases} \tag{12}$$

If the bicycle model has fallen over or driven far away from the path, which means that the control task is not solved, $e_i = 1$ is set. Otherwise, the error e_i is computed using the mean of t_0 measured over the last 80% of the duration of the simulated bicycle ride, denoted as $\bar{t}_{0,80\%}$. The first 20% of the bicycle ride are ignored for this computation, in order to make e_i independent of the perturbations the bicycle model may be initialized with. Remark that the distance t_0 is calculated the same as for the reward ρ, which is described in Sect. 3.3. An error threshold η_e is introduced, reading

$$\eta_\mathrm{e} = \frac{0.2\ \mathrm{m}}{\eta_\mathrm{y}} = \frac{0.2\ \mathrm{m}}{3.5\ \mathrm{m}} \approx 0.057\,. \tag{13}$$

In a validation process, a single simulated bicycle ride is called successful if $e_i \leq \eta_\mathrm{e}$ holds. Since a validation consists of 10 simulated bicycle rides, a validation results in 10 errors e_i.

4.3 Curriculum learning

Curriculum Learning (CL) is a training strategy for the field of machine learning that was first formalized in the work [36]. When doing CL, the task to be solved becomes increasingly complex as the learning proceeds in order to improve the quality of the learned policy and the speed of the learning process as compared to learning the complex task directly. CL is applied in the present work by changing the initialization interval of some parameters, marked in Table 3, throughout learning.

The adaptation of the initialization intervals is done using discrete stages, called parts of the curriculum. The functionality of the curriculum applied in the present work is schematically illustrated in Fig. 7. A linear relationship over $n_\mathrm{dec} = 10{,}000$ learning steps is used in

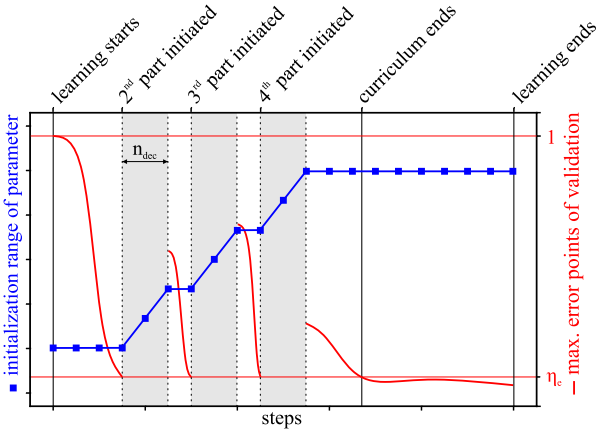

Fig. 7 Scheme of a four-part curriculum in which the initialization range of one parameter is adjusted. When a validation is fully successful, with all errors $e_i \leq \eta_e$, the next part of the curriculum is initiated. During changing the initialization range within n_{dec} learning steps, no validation is done. A fully successful validation in the last part of the curriculum represents the end of the curriculum

Table 4 Intervals in which the parameters that are changed throughout the learning process are initialized in each part of the four-part curriculum, where for each parameter x an interval $a \leq x \leq b$ is given

Parameter	Initialization range at curriculum part							
	1		2		3		4	
	a	b	a	b	a	b	a	b
forward velocity v in ms^{-1}	4	4	2	7	2	7	2	7
radius of circular elements R in m	14	14	12	14	10	14	8	14
length of polynomial elements L_{P} in m	20	22	18	22	16	22	14	22
width of polynomial elements W_{P} in m	2	2	2	4	2	6	2	8

the present work to extend the initialization intervals. In Table 4, the initialization intervals that are changed with the parts of the curriculum are given. It can be seen that the curriculum in the present work extends the forward velocity at which the bicycle is initialized from a constant value of 4 ms^{-1} to a wide range of speeds. The speed range goes from 2 ms^{-1}, a velocity at which the bicycle is in a highly unstable mode, via the self-stable velocity range from 4.3 ms^{-1} to 6 ms^{-1}, up to 7 ms^{-1}, where the bicycle is again mildly unstable [8]. This makes the task more complex in two ways, since the policy must get speed-dependent, see introductory Sect. 1, and the slower bicycle is more difficult to stabilize [8]. As the learning proceeds, the curriculum also makes the paths to be followed more complex by adding sharper turns and steeper transfers to the assembled paths.

5 Results

The section on the results is divided into five parts. Firstly, different settings for the RL framework are listed. Secondly, the learning process when using the different settings is analyzed. Afterwards, the performance of the learned controllers is evaluated. Then, a simulated bicycle ride along the benchmark path is analyzed in more detail. Finally, explanatory methods are applied to analyze the functioning of a learned controller. For more statistical robustness of the presented results, five learning processes, referred to as `runs`, initialized

Table 5 Variation of the RL framework. The single parameter that distinguishes the alt(1-5) settings from the proposed (prop.) settings is highlighted

Settings	$\mathbf{q} \vee \mathbf{q}'$	Δs	t_0	ρ_y	χ_1	Algorithm
prop.	$\mathbf{q}'$	v 0.4 s	$d(\mathrm{P}, \mathrm{P_{P0}})$	$\rho_\mathrm{y}^\mathrm{lin}$	1	SAC
alt(1)	$\mathbf{q}$	v 0.4 s	$d(\mathrm{P}, \mathrm{P_{P0}})$	$\rho_\mathrm{y}^\mathrm{lin}$	1	SAC
alt(2)	$\mathbf{q}'$	2 m	$d(\mathrm{P}, \mathrm{P_{P0}})$	$\rho_\mathrm{y}^\mathrm{lin}$	1	SAC
alt(3)	$\mathbf{q}'$	v 0.4 s	$d(\mathrm{Q}, \mathrm{P_{Q0}})$	$\rho_\mathrm{y}^\mathrm{lin}$	1	SAC
alt(4)	$\mathbf{q}'$	v 0.4 s	$d(\mathrm{P}, \mathrm{P_{P0}})$	$\rho_\mathrm{y}^\mathrm{exp}$	1	SAC
alt(5)	$\mathbf{q}'$	v 0.4 s	$d(\mathrm{P}, \mathrm{P_{P0}})$	$\rho_\mathrm{y}^\mathrm{lin}$	0.5	SAC

with a different random seed, are done under the investigated settings. Therefore, throughout this section, selection procedures are performed to start with a general analysis and get a single learned controller that is analyzed.

5.1 Investigated settings for the RL framework

The settings for the RL framework concern the observation vector $\mathbf{o}$, the design of the reward ρ, and the used RL algorithm.

In the observation vector $\mathbf{o}$, either the vector $\mathbf{q}$ for the minimal coordinates of the bicycle model, see Equation (1), or a redundant formulation $\mathbf{q}'$ with the yaw of the bicycle given as a unity vector, see Equation (4), is used. Furthermore, the preview distance Δs used to obtain the preview information is either set constant with $\Delta s = 2$ m or scaled with the forward velocity v of the bicycle, reading

$$\Delta s = v\ 0.4\ \mathrm{s}. \tag{14}$$

The distance t_0 used to compute the part ρ_y of the reward is either the distance $d(\mathrm{P}, \mathrm{P_{P0}})$ using the rear wheel contact point P, or the distance $d(\mathrm{Q}, \mathrm{P_{Q0}})$ using the front wheel contact point Q. With this distance, either a linear relationship $\rho_\mathrm{y}^\mathrm{lin}$, see Equation (8), or an exponential relationship $\rho_\mathrm{y}^\mathrm{exp}$, see Equation (9), can be used to compute the part ρ_y of the reward. The reward weights χ_1 and χ_2 are adjustable, see Equation (7). In the present work, either $\chi_1 = 1$ and $\chi_2 = 0$ or $\chi_1 = \chi_2 = 0.5$ is used. The latter setting is used to investigate how the inconsistency of the two reward parts ρ_y and ρ_φ, which is described in Sect. 3.3, disrupts the learning progress.

While any RL algorithm can be used in the introduced RL framework, as long as a continuous action space is supported by the algorithm, the SAC algorithm [31] performed significantly better in initial tests than the A2C algorithm introduced in the work [37] and the PPO algorithm introduced in the work [38], commonly referred to by their abbreviations. The results shown are therefore achieved exclusively with the SAC algorithm. The implementation of SAC in Stable-Baselines3 with the hyperparameters given in [31] is used. Variations in the network architecture of the networks used within the agent have not shown positive effects on the learning process, so the default settings from Stable-Baselines3 are used here as well.

In Table 5, the investigated settings are listed. Proposed settings are defined, to which each alternative combination of settings differs in one single parameter to better understand their influence on learning and the learned controllers.

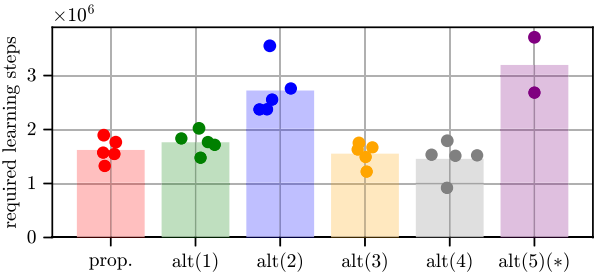

Fig. 8 The learning steps taken by the `runs` to complete the curriculum are indicated with dots for each RL framework setting. The mean of the learning steps, i.e. the required learning steps, is shown for each setting using a bar (∗ Not all of the `runs` done with these settings have reached the end of the curriculum)

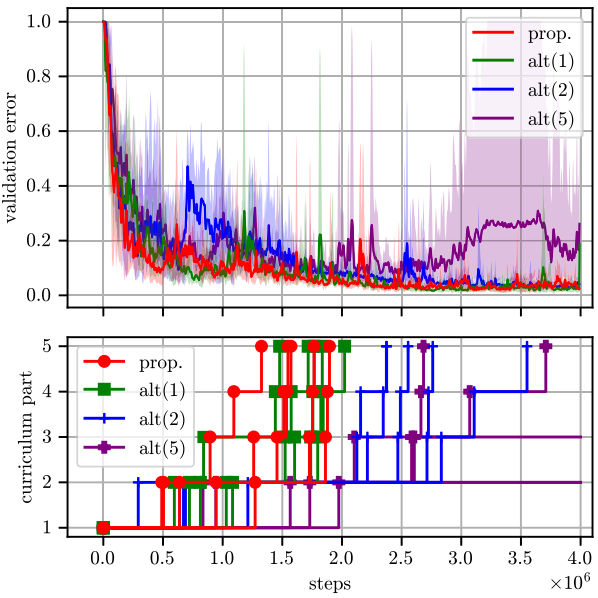

Fig. 9 (Top) Validation error e with the proposed and alt(1-2, 5) settings plotted over the learning steps. The mean validation error of the five `runs` done with each of the settings is drawn, as well as the area between the minimal and maximal validation error. A Simple Moving Average (SMA) filter with 20 data points is applied. (Bottom) Parts of the curriculum in which the individual `runs` with the different settings are, plotted over the learning steps. Part 5 corresponds to ending the curriculum in Fig. 7 (Color figure online)

5.2 Learning process

To analyze the learning process with the different settings for the RL framework, the required learning steps, the progression of the validation error over the learning steps, and the learning steps required in the parts of the curriculum are considered.

The number of learning steps required when using specific settings for the RL framework is the mean number of learning steps the `runs` with these settings take to end the curriculum. As can be seen in Fig. 8, the alt(2) settings, where the preview distance Δs is not scaled with the forward velocity v, and the alt(5) settings, where the reward weights χ_1 and χ_2 are set equally, require more than $2.5 \cdot 10^6$ learning steps. The proposed and alt(1, 3-4) settings require less than $2 \cdot 10^6$ learning steps. With regard to the alt(1) settings, this implies that the yaw representation does not influence the required learning steps.

As described in Sect. 4.2, each validation during the learning process results in 10 errors e_i. To obtain the learning progress, the validation error e, being the mean value of the errors e_i, is computed over the learning steps for each `run`. Thus, five errors e can be computed over the learning steps for each setting. In Fig. 9, the validation error e is plotted for the proposed and the alt(1-2, 5) settings over the learning steps, as well as the parts of the curriculum in which the `runs` with these settings are. No significant difference is observed

between the proposed and the alt(1) settings, where the yaw of the bicycle is given as an angle value. In the first part of the curriculum, the learning process with the proposed and the alt(2) settings with a non velocity-scaled preview distance Δs does not differ significantly, since the velocity scaling of Δs has no influence in the first part of the curriculum, with the forward velocity of the bicycle being constant, see Table 4. In the second part of the curriculum, the interval for the forward velocity v is extended and the learning processes with the alt(2) settings take an average of 264% of the learning steps taken when using the proposed settings. Furthermore, the mean validation error rises above 0.4 with the alt(2) settings. The learning with the alt(2) settings does not recover from the delay in the second part of the curriculum until the end of the learning process, necessitating the higher number of required learning steps already shown in Fig. 8. Two learning processes finish the curriculum in n_{tot} learning steps if the alt(5) settings with equally weighted reward parts are used. The other `runs` stay in the second and third part of the curriculum until n_{tot} learning steps are reached. The validation error starts to increase after $2.5 \cdot 10^6$ learning steps, implying that including the part ρ_φ in the reward disrupts the learning significantly. Learning processes with the alt(3) settings, where the front wheel contact point Q is used to obtain the distance t_0, and the alt(4) settings, where the exponential relationship $\rho_{\text{y}}^{\text{exp}}$ is used, show no significant difference from learning processes with the proposed or the alt(1) settings.

5.3 Performance of the learned controllers

In this section, the performance of the learned controllers is shown, as well as the influence the settings for the RL framework have on the performance of the learned controllers. Firstly, it is described at which point in a `run` the adaptation of the agent is stopped and the observation-action mapping is deployed as controller and how the performance is measured. Afterwards, the learned controllers are tested with the family of paths used during learning and along the benchmark path. At the end of this section, a learned controller is tested traversing a road intersection and thereby compared to two methods from the literature.

5.3.1 Controller selection and performance measure

The actor network, see Sect. 3 for more information about this network, of the agent is saved periodically during training. For performance evaluation, the most recent actor network that successfully completed all validation tests during training is deployed as learned controller. This ensures that the evaluated learned controllers represent the most advanced training state attainable within n_{tot} learning steps. Consequently, five learned controllers are evaluated each with the proposed and alt(1-4) settings and two learned controllers are evaluated with the alt(5) settings. During the simulated bicycle rides with the bicycle model being controlled by the learned controllers, the distance $t_0 = d(\text{P}, \text{P}_{\text{P0}})$ is tracked. Note that also for the alt(3) settings, where $d(\text{Q}, \text{P}_{\text{Q0}})$ is used to compute the reward, the distance $d(\text{P}, \text{P}_{\text{P0}})$ using the rear wheel contact point P is used here so that the results are comparable. The mean of the measured distance t_0 during the bicycle ride when using the learned controller k at a forward velocity of v is denoted as $\bar{t}_{0,\text{k}}(v)$. The maximum of the measured distance t_0 is denoted as $\hat{t}_{0,\text{k}}(v)$. Both values are used to quantify the performance of the learned controller k. Assuming that the learned controllers $k \in \mathbb{N} \cap [1, N]$ are deployed from the N `runs` with the proposed settings, the distance $T(v)$ reading

$$T(v) = \frac{1}{N} \sum_{k=1}^{N} \bar{t}_{0,\text{k}}(v) \tag{15}$$

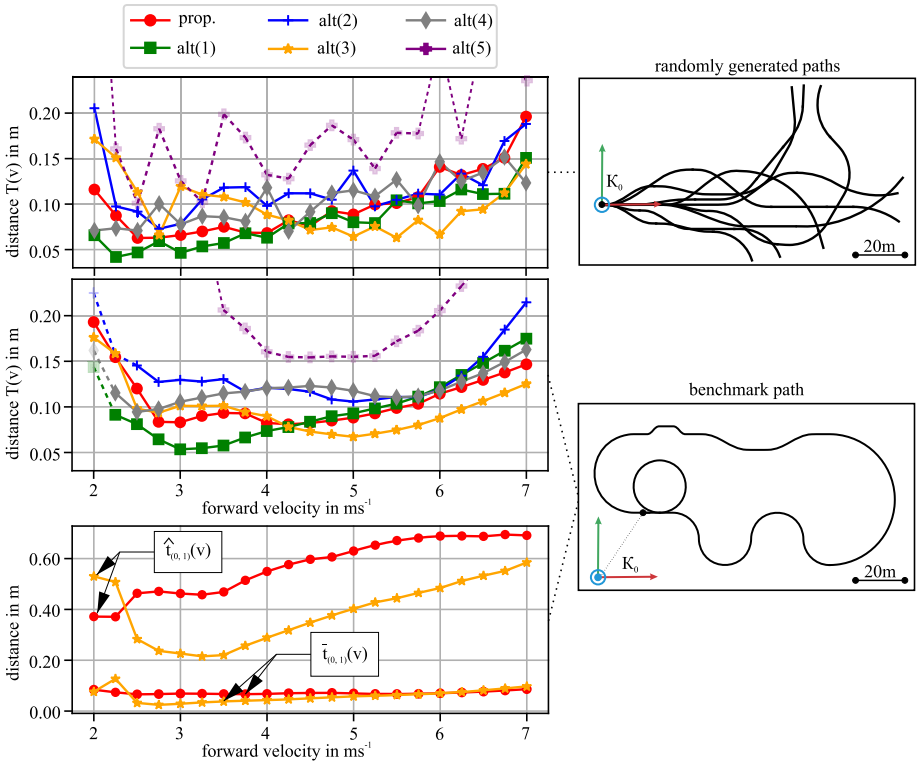

Fig. 10 (Top, Middle) Distance $T(v)$ plotted over the forward velocity v for the investigated settings when two different types of paths are used (schematic representation of the paths shown). (Bottom) Two individual learned controllers are analyzed in more detail. The mean $\bar{t}_{0,1}(v)$ and maximal distance $\hat{t}_{0,1}(v)$ are plotted, giving the performance of the two learned controllers over the investigated velocity range along the benchmark path

gives the overall performance of the proposed settings at the forward velocity of v. In the following, $T(v)$ is computed for all the settings shown in Table 5 with $2\ \mathrm{ms}^{-1} \leq v \leq 7\ \mathrm{ms}^{-1}$ applying for the forward velocity v. Note that during the simulated bicycle rides, the same observation settings apply as during learning. If, for example, an agent is trained with a non velocity-scaled preview distance, the resulting learned controller is also evaluated with the preview distance being non velocity-scaled.

5.3.2 Performance along randomly generated paths

The randomly generated paths for evaluation always start in the origin of the 0-frame and are assembled identically as in the last part of the curriculum by the reset procedure described in Sect. 4.1. The bicycle model is initialized in the reference configuration, see Fig. 1, with the rear wheel contact point P on the path. The duration of the simulated bicycle rides is set to 30 s. In the upper part of Fig. 10, the distance $T(v)$ is plotted for different settings. If one or more of the five learned controllers that can be deployed from one setting fails at a specific forward velocity v, the corresponding marker in the plot is shown translucent and connected by dotted lines. From the outset, this is the case for all markers of the alt(5) settings since only two agents finish the curriculum and thus only two learned controllers can be tested. It

Table 6 Maximal values for $\hat{t}_{0,1}(v)$ and $\bar{t}_{0,1}(v)$ when the two selected learned controllers are used to do path following and stabilization of the bicycle model along the benchmark path. The velocities at which the maximal values occur are given

Name	Value in m	Velocity v in ms^{-1}
`controller 1` trained using the proposed settings		
maximal value for $\hat{t}_{0,1}(v)$	0.69	7
maximal value for $\bar{t}_{0,1}(v)$	0.09	7
`controller 1` trained using the alt(3) settings		
maximal value for $\hat{t}_{0,1}(v)$	0.58	7
maximal value for $\bar{t}_{0,1}(v)$	0.13	2.25

is seen that all of the controllers learned using the proposed and the alt(1-4) settings are able to do path following and stabilization of the bicycle model along the randomly generated paths over the entire speed range. Taken together, for the evaluation of the proposed and alt(1-4) settings 525 simulated bicycle rides are made in total (21 velocities are tested with the learned controllers resulting from the five `runs` for each of the five settings) and in all 525 bicycle rides the bicycle is controlled successfully. In other words, none of the agents trained with the proposed and alt(1-4) settings results in a learned controller that fails in solving the control task for the type of paths the agents are trained with.

5.3.3 Performance along the benchmark path

The bicycle model is again initialized in the reference configuration, with the rear wheel contact point P standing at the starting point of the benchmark path, whose geometric properties are found in Fig. 5. The simulated bicycle rides along the benchmark path do not have a time limit, but the simulations stop when the bicycle returns to the start point. Figure 10 shows that all of the controllers learned using the proposed settings manage to do path following and stabilization of the bicycle model along the benchmark path over the entire velocity range. The same applies for the controllers learned using the alt(3) settings, where the distance $d(\mathrm{Q}, \mathrm{P}_{\mathrm{Q0}})$ with the front wheel contact point Q is used to compute the reward. The distance $T(v)$ when using the proposed settings is below 0.2 m over the entire speed range. At 4.25 ms^{-1} the distance T equals 0.08 m, being the smallest distance $T(v)$ for the proposed settings. It is found that at the lower end of the investigated velocity range, T reads 0.19 m and at the upper end T equals 0.15 m. For each of the alt(1, 4) settings, one learned controller does not succeed at 2 ms^{-1}. Two controllers learned using the alt(2) settings do not succeed at 2 ms^{-1} and one learned controller using the alt(2) settings fails at 2.25 ms^{-1}. This fact will be part of the closing discussion of the present work.

For each of the proposed and the alt(3) settings, one learned controller is now analyzed in more detail. To select the best learned controller among the five controllers obtained from the five runs under these two settings, the maximal distance $\hat{t}_{0,k}(v)$ for each controller k trained with these settings is computed. The value for $\hat{t}_{0,\mathrm{k}}(v)$ is maximal at a certain forward velocity. The learned controller with the smallest maximal value for $\hat{t}_{0,\mathrm{k}}(v)$ is selected. Doing so, for the proposed settings and the alt(3) settings one learned controller, called `controller 1` ($k = 1$) is chosen each. In the bottom part of Fig. 10, the distance values $\hat{t}_{0,1}(v)$ and $\bar{t}_{0,1}(v)$ are plotted over the velocity when these two learned controllers are used. Put plainly, the figure shows the performance of the controllers depending on the forward velocity of the bicycle, with the mean distance to the benchmark path and the maximal distance being given. Table 6 shows the maximal values of these two distances.

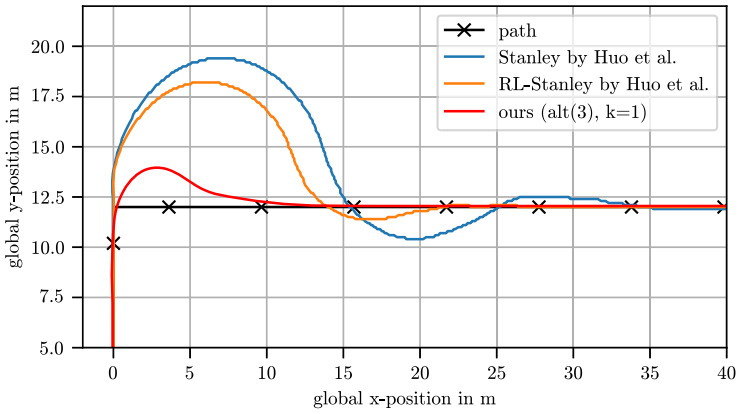

Fig. 11 Extreme curvature test of the here presented approach, the Stanley controller, and the RL-Stanley controller introduced in [23]. The front wheel contact points Q of the bicycles are tracked while the bicycle models move at 4 m s^{-1}. Note that in [23] a slightly different bicycle model is used than the one used in the present work, as it does not adopt the benchmark model from [8], see Sect. 2.1, and the present work uses preview points (Color figure online)

5.3.4 Performance under extreme curvature

The path shown in Fig. 11 represents a road intersection where the bicycle is supposed to turn right coming from below. This extreme curvature task is used in [23] to test the Stanley controller, but also the therein presented hybrid Stanley-RL approach, which is shortly described in the introductory Sect. 1 of the present work.

In the following, `controller 1` trained with the alt(3) settings is used to perform the same task. The alt(3) settings were selected because, as in [23], the front wheel contact point is shown in Fig. 11, substantiating choosing a controller learned using the Euclidean distance $d(\mathrm{Q}, \mathrm{P}_{\mathrm{Q0}})$. It is seen that the bicycle model controlled solely by the Stanley controller deviates from the desired path. Introducing an RL-based compensator improves path-following performance clearly. However, the fully RL-based approach from the present work, incorporating preview points, CL, and various framework adaptations, shows a significantly smaller deviation: While the Stanley-RL approach exhibits a maximum deviation of approximately 6 m, the method proposed in the present work remains within 2 m of the path.

Remember, that the agents have not seen the extreme curvature task during training, but the learned controller still manages to do path following along the path while stabilizing the bicycle model laterally. This fact will be part of the closing discussion of the present work. However, it should be noted beforehand that the purpose of this comparison is not to demonstrate that the purely RL based approach presented here is superior to other methods, as the approaches differ substantially. For instance, the approach presented here utilizes preview points, which is not the case for the method presented in [23], see also introductory Sect. 1.

5.4 Bicycle ride along the benchmark path in detail

In this section, a simulated bicycle ride along the benchmark path is analyzed in more detail, with the `controller 1` learned using the proposed settings controlling the bicycle model traveling at 4 ms^{-1}.

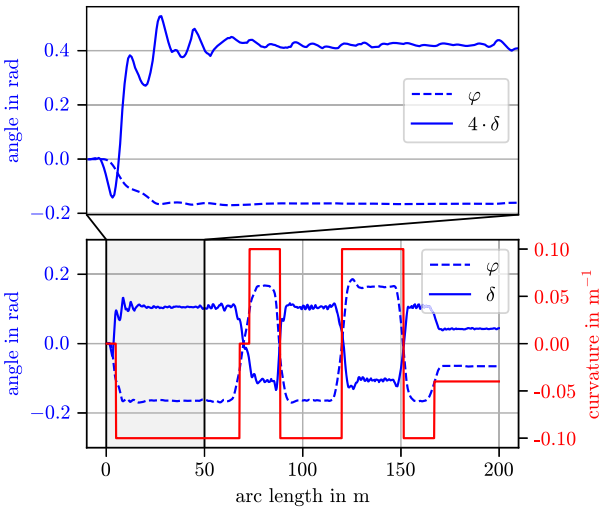

Fig. 12 Roll angle φ, steering angle δ, and curvature κ of the path plotted over the first 200 m of the benchmark path. `Controller 1` learned using the proposed settings is used to do path following and stabilization of the bicycle model. The bicycle model travels at 4 ms^{-1}

In Fig. 12, the roll angle φ, the steering angle δ, and the (negated) curvature κ of the benchmark path are plotted over the first 200 m of the benchmark path. It is seen that the roll angle φ of the bicycle model is proportional to the curvature κ of the path. In the zoomed-in area, it is seen that the steering angle δ is permanently adjusted by the learned controller and that δ gets negative, that is steering to the right, prior to the left curved circle. Such a counter-steering action is necessary to steer the bicycle in the desired direction [35], refer to Sect. 4.1. Figure 13, Frame (a) also shows this counter-steering action done before the bicycle follows the full circle. With the counter-steering, the roll angle φ gets negative. Along the full circle, a negative roll angle φ, induced by the prior counter-steering, is kept. The learned controller sets the steering angle δ positive, keeping the bicycle model upright by steering into the direction of the undesired fall [9], see also introductory Sect. 1, and doing path following along the circle. Frame (b) refers to a point where the curvature κ changes the sign. Frame (c) shows the bicycle traveling along the flat curve of the path. In Frame (d), the bicycle enters the first polynomial transfer. Frames (e) and (f) show the bicycle traveling along the so-called hard lane change. It is seen that the hard lane change is challenging to follow with the bicycle traveling at 4 ms^{-1}.

The supplementary material to the present work contains videos that show the `controller 1` learned using the proposed settings being used to do path following and stabilization of the bicycle model along the benchmark path, but also along other paths and at different forward velocities.

5.5 Explanation for the output of the learned controller

In Sect. 5.4, a simulated bicycle ride along the benchmark path using the `controller 1` trained with the proposed settings is described in detail. During the slalom part of the benchmark path, the observation vector $\mathbf{o}$ is saved every time the learned controller interacts with the environment so that observations are gained that equally represent the bicycle model doing path following along left and right curves. Using these observations, characteristics of the learned observation-action mapping are explained in this section using the SHapley Additive exPlanations (SHAP) values.

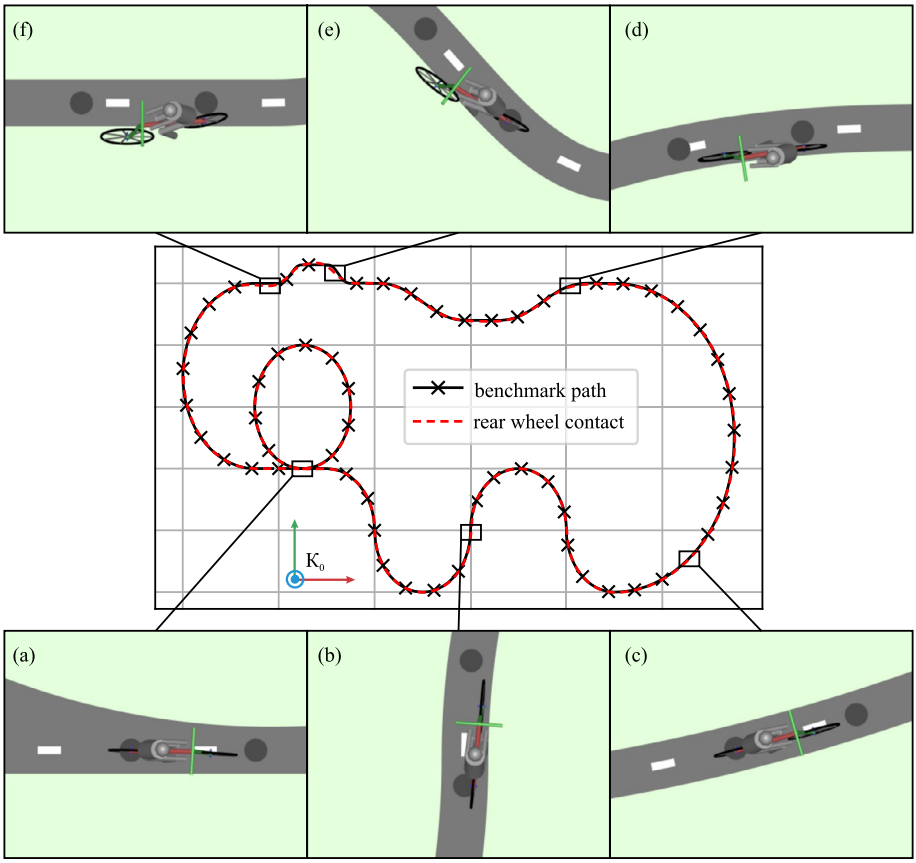

Fig. 13 Rear wheel contact point P tracked when the `controller 1` learned using the proposed settings is used to do path following and stabilization of the bicycle model along the benchmark path. Six different frames (a)-(f) from the visualization of the multibody simulation are shown at specific points of the simulated bicycle ride. The circles shown on the street illustrate the points $P_{P0}, \ldots, P_{P4}$ used to get the preview information of the path, see Fig. 6. The bicycle model travels at 4 ms^{-1}

SHAP values are a measure for the feature importance, introduced in the work [39]. The features are the elements in the input vector, being the elements of the observation vector $\mathbf{o}$. The permutation explainer, implemented in the eponymous Python package SHAP,[5] is used to compute the SHAP values. For each individual observation, being a saved observation vector $\mathbf{o}$, every feature can be assigned a SHAP value. A positive SHAP value of a feature in an observation means that the feature positively influences the output of the learned controller in that observation, while a negative SHAP value means that the feature negatively influences the output. Remember that the output of the learned controller, i.e. the output of the observation-action mapping, is the set value δ_{set} for the steering angle.

In Fig. 14, the importance of the features measured by the SHAP values is shown. The features are ordered by their mean absolute SHAP value. The roll angle φ is the feature having the greatest impact on the output of the learned controller. In the observations with a

[5] https://github.com/shap/shap (used version: 0.45.0).

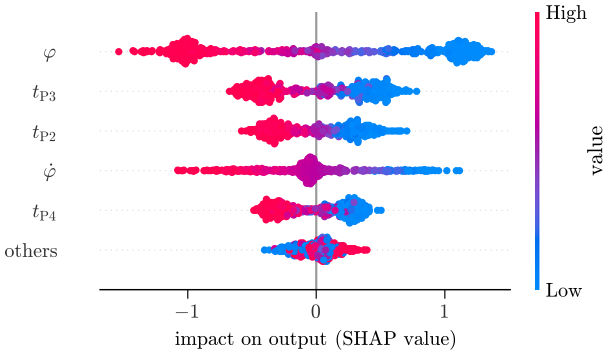

Fig. 14 Impact of the features on the output of a learned controller measured by the SHAP values. Each dot represents a single observation. The vertical alignment of the dot indicates whether the output is positively or negatively influenced by the corresponding feature in the observation. The color of the dot indicates whether the value of this feature is high or low in the observation (Color figure online)

positive roll angle φ, the roll angle φ contributes negatively to the output. Thus, a positive roll angle φ pulls the set value δ_{set} for the steering angle down and vice versa. The entry t_{P3} of the preview vector $\mathbf{t}_P$ is the second most important feature based on the mean absolute SHAP values. Observations with a positive value for t_{P3} negatively impact the output. The same applies to t_{P2}, but in the observations this feature does not contribute that much to the output of the learned controller as t_{P3}. There are observations, in which the roll angular velocity $\dot{\varphi}$ has a relatively high impact on the output. However, in most of the outputs, the roll angular velocity $\dot{\varphi}$ is near to zero, not influencing the set value δ_{set} for the steering angle. The mean absolute SHAP value of t_{P4}, which represents the preview point that is most ahead of the bicycle, is smaller than that of the roll angular velocity $\dot{\varphi}$. The other features are taken together as *others*, as they do not influence the output of the learned controller as much as the other features do, according to the SHAP values. The fact that x_Ψ and y_Ψ describing the yaw of the bicycle are part of *others* is in line with the finding that the yaw representation does not influence the learning process, see Sect. 5.2, since the yaw does not influence the output of the learned controller heavily.

6 Discussion

This section starts by discussing the different investigated settings for the RL framework. Afterwards, the results shown for the performance of the learned controllers are debated. Finally, the explanation for the outputs of a learned controller given by the SHAP values is compared to already known mechanisms for controlling a bicycle.

In Sect. 5.2, it is shown that using a constant preview distance Δs or including the part ρ_φ in the reward lowers the learning performance. On the former, we find that scaling Δs with the forward velocity reduces the learning volume. The learned policy must, because of using the Whipple bicycle, take the forward velocity into account to stabilize the bicycle laterally. The scaling of the preview distance eliminates another speed-dependence of the policy. What is striking at first is that the speed-dependence of the policy is not represented by the SHAP values, see Fig. 14, since the front wheel rotation angular velocity $\dot{\theta}_F$ is classified into *others*. The reason for this is not the lack of speed-dependence in the policy, but the way the SHAP values are computed. In the simulated bicycle ride used to obtain the observations for the explanation, the bicycle travels at a constant forward velocity of 4 ms^{-1}, see Sect. 5.5. Thus, the permutation explainer iterates over almost constant values for the front wheel rotation angular velocity $\dot{\theta}_F$ with the output of the learned controller not changing within this iteration. Regarding the reward design, we discuss whether including the roll

angle φ in a more sophisticated way than using the deviation from zero, see Equation (10), would make the consideration of the part ρ_φ beneficial for the learning progress, i.e. using the deviation from an optimal roll angle that must be computed for the current position of the bicycle. We also think of using the roll angular velocity $\dot{\varphi}$ and a threshold $\eta_{\dot{\varphi}}$ instead of the roll angle φ and the roll angle threshold η_φ in Equation (10). Initial tests with $\chi_1 = \chi_2 = 0.5$ show that when using $\dot{\varphi}$ the learning improves in comparison to that with the alt(5) settings, but is still worse than when only using ρ_y.

In Sect. 5.3, it is shown that all controllers learned using the proposed and alt(1-4) settings succeed along the type of paths that is used during the learning process. However, individual learned controllers fail along the benchmark path at the lower end of the investigated range for the forward velocity. To explain this, we have a look at the bicycle rides along the benchmark path and find that the individual learned controllers fail at two specific segments of the benchmark path, namely the full circle and the hard lane change. We suspect that these two segments are not part of the learning process and therefore cause difficulties, which we want to demonstrate in the following. Due to the geometric properties of the hard lane change, see Fig. 5, the hard lane change does not occur in the paths during learning, see Table 4. To estimate the probability P of a full circle being part of a single path during learning, it is assumed that for the bicycle traveling at 2 ms^{-1} the path is 120 m long, resulting from n_max that is specified in Sect. 4. In the first step, 10^4 paths are generated at random using the reset procedure described in Sect. 4.1 and the initialization intervals from the last part of the curriculum. Using the expected value for the uniformly distributed angle Φ of a circular element, a full circle is made out of approximately five consecutive circular elements with the same curvature. In the randomly generated paths, 20 paths have five circular elements in a row with the same curvature, resulting in $P = 20/10^4 = 0.002$ for the probability of a full circle. The two segments are therefore most likely neither part of the learning process nor of the validation process, which, in combination with a low forward velocity, leads to individual learned controllers failing. The speed-dependence is due to the increasing difficulty of stabilizing the bicycle model as velocity decreases and the real part of the Whipple bicycle's eigenvalues increases [8]. On the one hand, the results for the benchmark path show that segments that are not part of paths used during the learning process can cause difficulties for the learned controllers, in particular at low forward velocities. On the other hand, the results show that most of the learned controllers have induced a policy with which path following and stabilization of the bicycle model can be done at all investigated forward velocities along a path that includes two segments that the agents statistically have not seen during learning, indicating a form of generalization capabilities. Looking, for example, at the `controller 1` trained with the proposed settings, a maximal mean distance of 0.09 m is found along the benchmark path at 7 ms^{-1}, being 9% of the wheelbase of the bicycle model. We feel that this is very accurate, considering that the benchmark path cannot be followed exactly due to the step-like curvature changes along the path and the hard lane change that is challenging to follow especially at higher forward velocities. The generalization capabilities are further supported by the fact that learned controllers, we have tested `controller 1` trained with the proposed and alt(3) settings, can manage the extreme curvature test, see Sect. 5.3.4, which is not part of the learning process.

In Sect. 5.5, SHAP values are used to explain the output of a learned controller. For a better understanding of this explanation, we discuss here a scenario in which the bicycle model is traveling on a straight and is to turn left. At first, the elements of the preview vector take on positive values. A combination of three of these elements, with $t_{\mathrm{P}3}$ being considered the most, see Fig. 14, negatively influences the outputted set value for the steering angle. Thus, the learned controller does not steer the bicycle to the left, but to the right. The learned

controller does counter-steering, which is what we expect the agent to learn, as explained in Sect. 4.1. With this counter-steering, the contact points P and Q move to the right, the roll angle φ gets negative, and the bicycle starts to lean into the left turn. As the agent learns to steer into the direction of the undesired fall, see introductory Sect. 1, a negative roll angle φ pulls the output of the learned controller up, which means that the handlebar is turned to the left, as the bicycle leans to the left. The steering angle δ gets positive and the leaned bicycle follows the left turn. We find that the learned controller also respects the roll angular velocity $\dot{\varphi}$, which is suspected, as some bicycle controllers use $\dot{\varphi}$ as controlled variable, see for example the intuitive approach described in [27]. Taken together, in the learned controller we find the two fundamental mechanisms of controlling bicycles considering the SHAP values, that is, to steer into the direction of the undesired fall and to do counter-steering.

7 Conclusion

Before future research is addressed, the present work is concluded by five key statements:

- It is demonstrated that a purely Reinforcement Learning (RL) based approach can be used to do path following with the Whipple bicycle while simultaneously stabilizing it laterally. No stabilization aids are needed. The agent learns to do path following and stabilization of the bicycle model exclusively by setting the steering angle.
- It is shown that the RL approach works for a wide range of forward velocities and that a speed-dependent policy is learned. The bicycle is presented as an example of RL being successfully applied to a regime-dependent control problem.
- Curriculum learning as a training strategy is used to make the RL approach applicable for the wide range of forward velocities and arbitrary paths.
- A learned controller resulting of an agent is presented, being able to steer the bicycle model along a benchmark path with a maximum mean distance of 0.09 m over the entire velocity range from 2 ms^{-1} to 7 ms^{-1}. The benchmark path consists of a full circle, a slalom, curves, two lane changes, and straight elements.
- The explanation for the output of the learned controller shows that basic mechanisms for controlling bicycles are learned. This builds a bridge to research in the field of bicycle dynamics.

Future research might address the practical realization of an intelligent bicycle, where a learned controller does path following and stabilization of the bicycle while its forward velocity is set by the cyclist. Here, above all, the modeling of the bicycle must be refined in order to facilitate the Sim-to-Real transfer, explained in the work [40] for RL in the field of robotics. The adaptation of the model concerns not only the bicycle model itself, but also the path on which the bicycle travels, being, for example, uneven. In addition, future work must consider a way of obtaining the preview information of the path. Possible approaches might be camera systems or the usage of mapped cycling routes. Furthermore, future academic works can use the benchmark path shown in Fig. 5 to compare the performance of bicycle controllers with the approach presented here or with one another.

Appendix A: Coordinates mappings for the whipple bicycle

In this section, it is shown how the output of a multibody simulation of the Whipple bicycle, introduced in [25], can be mapped to the minimal coordinates of the bicycle model. Afterwards, the mapping from the bicycle's minimal coordinates to the redundant coordinates

Table 7 Notation and spatial transformations used for the coordinates mappings

Notation	Description
$\mathbf{0}$	zero matrix (or zero vector)
$\mathbf{1}$	unity matrix
${}^{\mathrm{K}}\mathbf{r}_{\mathrm{P_1P_2}}$	vector pointing from the point $\mathrm{P_1}$ to the point $\mathrm{P_2}$ with its coordinates formulated in the K-frame
${}^{\mathrm{KM}}\mathbf{R}$	rotation that rotates coordinates of vectors formulated in the M-frame into the K-frame: ${}^{\mathrm{K}}\mathbf{r} = {}^{\mathrm{KM}}\mathbf{R} \cdot {}^{\mathrm{M}}\mathbf{r}$
${}^{\mathrm{KM}}\mathbf{T}$	homogeneous transformation that transforms coordinates of vectors formulated in the M-frame into the K-frame
$\widetilde{\mathbf{a}}$	skew symmetric matrix of the vector $\mathbf{a}$; used to calculate the cross product $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ as a matrix vector multiplication: $\mathbf{c} = \widetilde{\mathbf{a}} \cdot \mathbf{b}$
$\mathrm{Rot_e}\,(\alpha)$	rotation by α around the unity vector $\mathbf{e}$

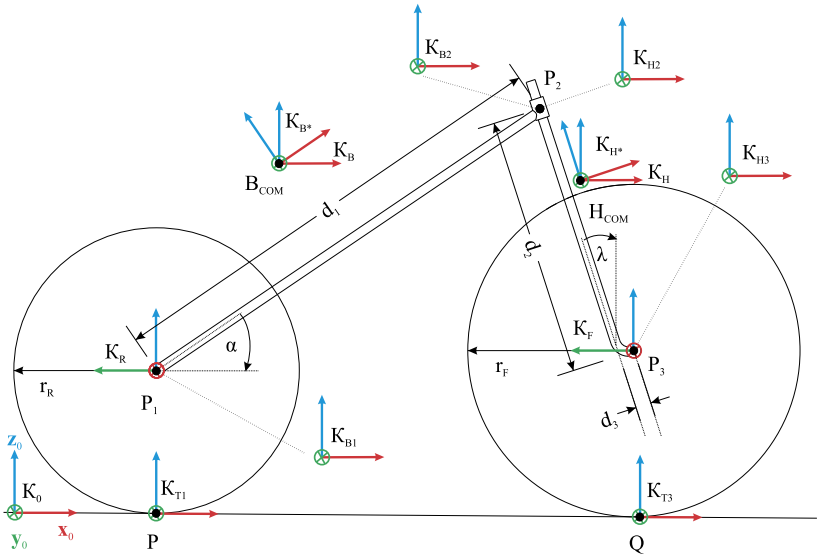

Fig. 15 Whipple bicycle shown in the reference configuration, with characteristic geometric parameters and coordinate systems that are needed for the coordinates mappings. Each coordinate system $\mathcal{K}_i$ is described by its origin and unity vectors $\mathbf{x}_i$, $\mathbf{y}_i$, and $\mathbf{z}_i$

is described. The relation between the minimal coordinates and the redundant coordinates on both the positional and velocity base of this non-holonomic system is shown in detail. In Table 7, the notation used in this section is given. The reference frames and geometric properties drawn in Fig. 15 are used in the following.

The global 0-frame, the $\mathrm{T_1}$-frame, and the $\mathrm{T_3}$-frame lie in the ground plane. The origin of the latter two frames each lies in the contact point of the wheel and its z-axis points towards the respective wheel hub. The R-, B-, H-, and F-frame are body-fixed frames of the eponymous bodies with their origin each being located at the COM of the rigid body. The R- and the F-frame are defined in such a way that their x-axes are the rotation axes of the wheels. The B- and the H-frame can each be translated to two further points on the

corresponding rigid body which is indicated with the subscript 1 and 2 for the rear body and with 2 and 3 for the handlebar. Two reference frames are marked with a star. They are obtained by rotating the B-frame by the negated frame angle α around its y-axis and the H-frame by the negated steering axis tilt λ around its y-axis. In the following mappings, also the two wheel radii r_R and r_F, the frame length d_1, the fork length d_2, and the fork offset d_3 are used.

A.1 Mapping of redundant to minimal coordinates using sensors

A clear explanation of the minimal coordinates of the Whipple bicycle can be found in [8]. The specific minimal coordinates used in the present work are those described in the main text of the article. To calculate the minimal coordinates, sensors are defined in the interface of the multibody simulation code. One sensor measures the position of the contact point P and thus provides the minimal coordinates x_P and y_P. Another sensor gives the rotation matrix ${}^{0B}\mathbf{R}$ describing the spatial orientation of the rear body in the global frame. Since the global frame can be turned into the B-frame by performing a sequence of turns

$$ {}^{0B}\mathbf{R} = \mathrm{Rot_z}\,(\Psi) \cdot \mathrm{Rot_x}\,(\varphi) \cdot \mathrm{Rot_y}\,(\theta_B)\,, \tag{A.1} $$

the yaw angle Ψ and the roll angle φ can be computed using the equations given in [41]. The angle θ_B is the pitch of the frame and not further needed for the mapping of redundant to minimal coordinates. The steering angle δ, the rear wheel rotation angle θ_R, and the front wheel rotation angle θ_F are output variables of sensors measuring the corresponding revolute joint angles. The roll angular velocity $\dot{\varphi}$ is the component of the rear body angular velocity in the direction of the x-axis of the T_1-frame. Thus, it can be written as the inner product

$$ \dot{\varphi} = \langle {}^0\mathbf{x}_{T_1}, {}^0\boldsymbol{\omega}_B \rangle, \tag{A.2} $$

where ${}^0\boldsymbol{\omega}_B$ is the output quantity of a sensor attached to the rear body. The x-axis ${}^0\mathbf{x}_{T_1}$ is known via a homogeneous transformation matrix reading

$$ {}^{0T_1}\mathbf{T} = \underbrace{\begin{bmatrix} \mathrm{Rot_z}\,(\Psi) & {}^0\mathbf{r}_{0P} \\ \mathbf{0} & 1 \end{bmatrix}}_{{}^{0C}\mathbf{T}} \cdot \begin{bmatrix} \mathrm{Rot_x}\,(\varphi) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{A.3} $$

The remaining two minimal coordinates, namely the steering angular velocity $\dot{\delta}$ and the front wheel rotation angular velocity $\dot{\theta}_F$ are measured by sensors attached to the corresponding revolute joint.

A.2 Mapping of minimal to redundant coordinates

In the following, it is shown how the redundant coordinates on positional and velocity base of the four rigid bodies the Whipple bicycle consist of can be computed from the minimal coordinates of the Whipple bicycle.

At first, four homogeneous transformation matrices are defined, representing the position and orientation of the bodies. The pose of the rear body is given by

$$ {}^{0B}\mathbf{T} = \underbrace{{}^{0T_1}\mathbf{T} \cdot \begin{bmatrix} \mathrm{Rot_y}\,(\theta_B) & {}^{T_1}\mathbf{r}_{PP_1} \\ \mathbf{0} & 1 \end{bmatrix}}_{{}^{0B_1}\mathbf{T}} \cdot \begin{bmatrix} \mathbf{1} & {}^{B}\mathbf{r}_{P_1B_{COM}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{A.4} $$

${}^{0\mathrm{T}_1}\mathbf{T}$ follows from equation (A.3). The pitch angle θ_B satisfies the fact that the rear body can pitch, which means that it can rotate around its y-axis. In [42], it is shown that an analytical solution for θ_B given the bicycle minimal coordinates exists. In [43], a geometric way to derive a fourth degree polynomial in θ_B is introduced. In Sect. C, a convenient way using rotation matrices to get the polynomial in θ_B and thus θ_B is derived. The rear wheel is represented with

$$ {}^{0\mathrm{R}}\mathbf{T} = {}^{0\mathrm{B}_1}\mathbf{T} \cdot \begin{bmatrix} \mathrm{Rot_z}\left(\frac{\pi}{2}\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{Rot_x}\left(\theta_\mathrm{R}\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} . \tag{A.5} $$

The transformation ${}^{0\mathrm{B}_1}\mathbf{T}$ used therein results from Equation (A.4). Using Equation (A.4), also the handle can be represented as follows:

$$ {}^{0\mathrm{H}}\mathbf{T} = {}^{0\mathrm{B}}\mathbf{T} \cdot \begin{bmatrix} \mathrm{Rot_y}\left(-\lambda\right) & {}^{\mathrm{B}}\mathbf{r}_{\mathrm{B_{COM}H_{COM}}} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{Rot_z}\left(\delta\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{Rot_y}\left(\lambda\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} . \tag{A.6} $$

The pose of the front wheel reads

$$ {}^{0\mathrm{F}}\mathbf{T} = {}^{0\mathrm{H}}\mathbf{T} \cdot \begin{bmatrix} \mathrm{Rot_z}\left(\frac{\pi}{2}\right) & {}^{\mathrm{H}}\mathbf{r}_{\mathrm{H_{COM}P_3}} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{Rot_x}\left(\theta_\mathrm{F}\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} , \tag{A.7} $$

where ${}^{0\mathrm{H}}\mathbf{T}$ is taken from Equation (A.6).

Furthermore, the translational and angular velocities of the rigid bodies must be computed, using the bicycle's minimal coordinates. The following sequence of computations is followed in the present work:

1. angular velocity of B-frame
2. angular velocity of H-frame
3. angular velocity of F-frame
4. translational velocity of F-frame
5. translational velocity of H-frame
6. translational velocity of B-frame
7. translational velocity of R-frame
8. angular velocity of R-frame.

It is started with the angular velocity of the rear body, that reads

$$ {}^{0}\boldsymbol{\omega}_\mathrm{B} = \begin{bmatrix} 0 \\ 0 \\ \dot{\Psi} \end{bmatrix} + {}^{0\mathrm{T}_1}\mathbf{R} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta}_\mathrm{B} \\ 0 \end{bmatrix} . \tag{A.8} $$

In Sect. C, it is described how the pitch angular velocity $\dot{\theta}_\mathrm{B}$ can be computed using rotation matrices. The yaw angular velocity $\dot{\Psi}$ results from

$$ \dot{\Psi} = \frac{1}{w} \sin\left(\mu_1 - \Psi\right) v \,, \tag{A.9} $$

where the forward velocity v of the bicycle is computed using the front wheel rotation angular velocity $\dot{\theta}_\mathrm{F}$ and the radius $\mathrm{r_F}$ of the front wheel: $v = \mathrm{r_F} \cdot \dot{\theta}_\mathrm{F}$. The wheelbase w of the bicycle is computed using Equation (C.6). The angle μ_1 describes the direction of motion of the front wheel contact point Q. Since the rotation matrix ${}^{0\mathrm{F}}\mathbf{R}$ is already known from Equation (A.7), this angle can be calculated by taking

$$ {}^{0\mathrm{F}}\mathbf{R} = \mathrm{Rot_z}\left(\mu_1\right) \cdot \mathrm{Rot_x}\left(\mu_2\right) \cdot \mathrm{Rot_y}\left(\mu_3\right) \tag{A.10} $$

into account, using the formulas shown in [41]. The angles μ_2 and μ_3 are not needed. The angular velocity of the handle is computed using the relative angular velocity between the rear body and the handle which results exclusively from the revolute joint connecting the rear body and the handle. It follows

$$ {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BH}} = \mathrm{Rot}_{\mathrm{y}}\left(-\lambda\right) \begin{bmatrix} 0 \\ 0 \\ \dot{\delta} \end{bmatrix} , \tag{A.11} $$

with λ being the steering axis tilt. Finally, for the handle applies

$$ {}^{0}\boldsymbol{\omega}_{\mathrm{H}} = {}^{0\mathrm{B}}\mathbf{R} \left({}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}} + {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BH}} \right) . \tag{A.12} $$

The same procedure is used to get the angular velocity of the front wheel

$$ {}^{0}\boldsymbol{\omega}_{\mathrm{F}} = {}^{0\mathrm{H}}\mathbf{R} \left({}^{\mathrm{H}}\boldsymbol{\omega}_{\mathrm{H}} + {}^{\mathrm{H}}\boldsymbol{\omega}_{\mathrm{HF}} \right) , \tag{A.13} $$

where the relative angular velocity between the handle and the wheel results from the front wheel hub and reads

$$ {}^{\mathrm{H}}\boldsymbol{\omega}_{\mathrm{HF}} = \begin{bmatrix} 0 & \dot{\theta}_{\mathrm{F}} & 0 \end{bmatrix}^{\mathrm{T}} . \tag{A.14} $$

The translational velocity of the front wheel follows with Euler's first theorem for kinematics to

$$ {}^{0}\mathbf{v}_{\mathrm{F}} = {}^{0}\mathbf{v}_{\mathrm{H}_3} = {}^{0}\mathbf{v}_{\mathrm{T}_3} + {}^{0}\widetilde{\boldsymbol{\omega}}_{\mathrm{T}_3} \, {}^{0}\mathbf{r}_{\mathrm{QP}_3} . \tag{A.15} $$

The therein used velocity of the front wheel contact point Q, that is ${}^{0}\mathbf{v}_{\mathrm{T}_3}$, is computed using the angle μ_1 and the bicycle's forward velocity v.

$$ {}^{0}\mathbf{v}_{\mathrm{T}_3} = v \begin{bmatrix} \cos\mu_1 \\ \sin\mu_1 \\ 0 \end{bmatrix} \tag{A.16} $$

Remember that v is computed using the front wheel rotation angular velocity $\dot{\theta}_{\mathrm{F}}$. The angular velocity of the T_3-frame needed for Equation (A.15) results from that of the front wheel, but without its rotation angular velocity:

$$ {}^{0}\boldsymbol{\omega}_{\mathrm{T}_3} = {}^{0\mathrm{T}_3}\mathbf{R} \left({}^{\mathrm{T}_3 0}\mathbf{R} \, {}^{0}\boldsymbol{\omega}_{\mathrm{F}} + \begin{bmatrix} 0 \\ -\dot{\theta}_{\mathrm{F}} \\ 0 \end{bmatrix} \right) . \tag{A.17} $$

Now, ${}^{0}\mathbf{v}_{\mathrm{F}}$ can be computed and it is continued with the velocity of the handle, reading

$$ {}^{0}\mathbf{v}_{\mathrm{H}} = {}^{0}\mathbf{v}_{\mathrm{H}_3} + {}^{0}\widetilde{\boldsymbol{\omega}}_{\mathrm{H}} \, {}^{0}\mathbf{r}_{\mathrm{P}_3\mathrm{H}_{\mathrm{COM}}} . \tag{A.18} $$

With the translational velocity of the bicycle's head

$$ {}^{0}\mathbf{v}_{\mathrm{B}_2} = {}^{0}\mathbf{v}_{\mathrm{H}_2} = {}^{0}\mathbf{v}_{\mathrm{H}} + {}^{0}\widetilde{\boldsymbol{\omega}}_{\mathrm{H}} \, {}^{0}\mathbf{r}_{\mathrm{H}_{\mathrm{COM}}\mathrm{P}_2} , \tag{A.19} $$

the one of the rear body follows to

$$^{0}\mathbf{v}_{\mathrm{B}} = {}^{0}\mathbf{v}_{\mathrm{B}_2} + {}^{0}\widetilde{\boldsymbol{\omega}}_{\mathrm{B}}\, {}^{0}\mathbf{r}_{\mathrm{P}_2\mathrm{B}_{\mathrm{COM}}} . \tag{A.20}$$

The translational velocity of the rear wheel reads

$$^{0}\mathbf{v}_{\mathrm{R}} = {}^{0}\mathbf{v}_{\mathrm{B}_1} = {}^{0}\mathbf{v}_{\mathrm{B}} + {}^{0}\boldsymbol{\omega}_{\mathrm{B}}\, {}^{0}\mathbf{r}_{\mathrm{B}_{\mathrm{COM}}\mathrm{P}_1} . \tag{A.21}$$

Finally, the angular velocity of the rear wheel can be expressed by

$$^{0}\boldsymbol{\omega}_{\mathrm{R}} = {}^{0\mathrm{B}}\mathbf{R} \left({}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}} + {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BR}} \right) . \tag{A.22}$$

The relative angular velocity between the rear body and the rear wheel results from the rear wheel hub. It is denoted as

$$^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BR}} = \begin{bmatrix} 0 & \dot{\theta}_R & 0 \end{bmatrix}^{\mathrm{T}} . \tag{A.23}$$

Since the therein used rear wheel rotation angular velocity is not an element of the minimal coordinates of the bicycle, it must be computed. Using the angular velocity of the T_1-frame that reads

$$^{\mathrm{T}_1}\boldsymbol{\omega}_{\mathrm{T}_1} = {}^{\mathrm{T}_1 0}\mathbf{R}\, {}^{0}\boldsymbol{\omega}_{\mathrm{B}} + \begin{bmatrix} 0 \\ -\dot{\theta}_{\mathrm{B}} \\ 0 \end{bmatrix} , \tag{A.24}$$

the translational velocity of it in its own frame can be computed as shown below.

$$^{\mathrm{T}_1}\mathbf{v}_{\mathrm{T}_1} = {}^{\mathrm{T}_1}\mathbf{v}_{\mathrm{R}} + {}^{\mathrm{T}_1}\widetilde{\boldsymbol{\omega}}_{\mathrm{T}_1}\, {}^{\mathrm{T}_1}\mathbf{r}_{\mathrm{P}_1\mathrm{P}} \tag{A.25}$$

Only the x-component of this vector is different from zero, which results from the fact that the wheels are slip free to the side. Consequently, the absolute velocity of the rear contact point P reads $v_{\mathrm{T}_1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot {}^{\mathrm{T}_1}\mathbf{v}_{\mathrm{T}_1}$. The velocity v_{T_1} only results from the rear wheel rotational angular velocity $\dot{\theta}_{\mathrm{R}}$. Thus, the rear wheel rotation angular velocity can be computed with

$$\dot{\theta}_{\mathrm{R}} = \frac{v_{\mathrm{T}_1}}{\mathrm{r}_{\mathrm{R}}} , \tag{A.26}$$

where the radius r_{F} of the rear wheel is used. This completes the mapping from the minimal to the redundant coordinates.

Appendix B: Threshold value for the steering angle of the whipple bicycle and step response of the steering angle

In this section, a threshold value for the steering angle δ of the Whipple bicycle is defined. The threshold value is used to define a range in which the set value δ_{set} for the steering angle the agent can output must be. With this threshold value, the step response of the steering angle δ is shown, which may be used to design controllers for the bicycle model, such as a PD controller mapping a set value δ_{set} for the steering angle to a steering torque.

Table 8 Tabular representation of the dependency between the roll angle φ, the steering angle δ, and the direction of motion μ_1 of the front wheel contact point Q of the Whipple bicycle. The minimal coordinates except of φ and δ of the Whipple bicycle are zeroed

Roll angle φ	Steering angle δ	Angle μ_1
0°	0°	0°
0°	60°	58.8°
0°	70°	69.11°
−45°	0°	0°
−45°	60°	81.66°
−45°	70°	92.16°
45°	0°	0°
45°	−60°	−81.66°
45°	−70°	−92.16°

As a result of the tilted steering axis and the fork offset of the Whipple bicycle, the direction of motion of the front wheel contact point Q relative to the T_1-frame is not equal to the steering angle δ of the Whipple bicycle, see Table 8. In the table, it can be seen that the direction of motion of the front wheel contact point, given by the angle μ_1, deviates significantly from the steering angle δ, especially in configurations of the Whipple bicycle where $|\varphi| \gg 0$ applies. The angle μ_1 is computed using Equation (A.10). Since the magnitude of the roll angle of the bicycle model in the present work should not exceed the threshold $\eta_\varphi = 45°$ and the direction of motion of the front wheel contact point Q must not exceed $\pm 90°$, a limit for the steering angle δ can be defined. It is assumed that limiting the steering angle δ to an value of $\pm 70°$ is sufficient (although $|\mu_1|$ slightly exceeds 90° when the bicycle is rolled by $|\varphi| = 45°$). Consequently, the set value δ_{set} the agent can output for the steering angle must be in the interval of $-70°$ to 70°.

The step response of the steering angle δ is used to determine the two parameters P and D of a PD controller that gives the torque τ that is imprinted between the rear body and the fork of the bicycle model. To get the step response, the stationary bicycle model is simulated when the set value δ_{set} for the steering angle follows

$$\delta_{\text{set}}(t) = \begin{cases} 0° & t < 0 \\ 70° & t \geq 0\,. \end{cases} \tag{B.1}$$

For this particular simulation, the gravitational acceleration is zeroed so that the bicycle does not fall over. In Fig. 16, the step response of the steering angle δ is shown. It is defined that the two parameters P and D should be set in a way so that the maximal value of the step response does not overshoot the set value by more than 10%. By setting $P = 9$ Nm and $D = 1.6$ Nms the specified criterion is fulfilled.

Appendix C: Pitch angle and pitch angular velocity of the whipple bicycle with rotation matrices

In the following, a way to compute the pitch angle θ_B of the Whipple bicycle's rear body using rotation matrices is shown, when the minimal coordinates of the bicycle model are given. The notation and reference frames that are used in Sect. A are also used in the following.

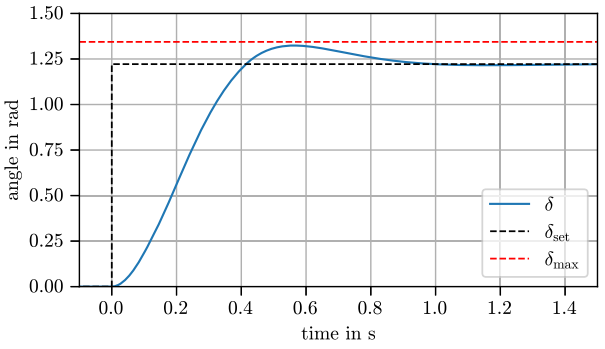

Fig. 16 Step response of the steering angle δ with the bicycle being stationary when the set value δ_{set} for the steering angle follows the function shown. To obtain the step response, the gravitational acceleration is zeroed. The defined maximal permissible value δ_{max} for the step response is also shown

It is started by getting the z-axis ${}^{0}\mathbf{z}_{\text{T}_1}$ from the T_1-frame formulated in the global 0-frame. This vector is the last column of the rotation matrix

$$ {}^{0\text{T}_1}\mathbf{R} = \text{Rot}_{\text{z}}(\Psi) \cdot \text{Rot}_{\text{x}}(\varphi) \,. \tag{C.1} $$

To rotate the 0-frame into the B*-frame, the rotation matrix

$$ {}^{0\text{B}^*}\mathbf{R} = \underbrace{{}^{0\text{T}_1}\mathbf{R} \cdot \text{Rot}_{\text{y}}(\theta_{\text{B}})}_{{}^{0\text{B}}\mathbf{R}} \cdot \text{Rot}_{\text{y}}(-\alpha) \tag{C.2} $$

is used, with the pitch angle θ_{B} being unknown. The x-axis ${}^{0}\mathbf{x}_{\text{B}^*}$ is the first column of this matrix. The rotation matrix for the H*-frame follows

$$ {}^{0\text{H}^*}\mathbf{R} = {}^{0\text{B}}\mathbf{R} \cdot \underbrace{\text{Rot}_{\text{y}}(-\lambda) \cdot \text{Rot}_{\text{z}}(\delta)}_{{}^{\text{BH}^*}\mathbf{R}} \,. \tag{C.3} $$

The axes ${}^{0}\mathbf{x}_{\text{H}^*}$, ${}^{0}\mathbf{y}_{\text{H}^*}$, and ${}^{0}\mathbf{z}_{\text{H}^*}$ are the columns of this matrix. As shown in [43], it is possible to calculate a unity vector ${}^{0}\mathbf{h}$ that points from the front wheel hub in the direction of Q by performing two steps:

$$ {}^{0}\mathbf{f} = -\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \left({}^{0}\mathbf{y}_{\text{H}^*} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) , \qquad {}^{0}\mathbf{h} = \frac{{}^{0}\mathbf{f}}{|\mathbf{f}|} \,. \tag{C.4} $$

A vector pointing from P to Q can be computed by

$$ {}^{0}\mathbf{r}_{\text{PQ}} = \text{r}_{\text{R}}\,{}^{0}\mathbf{z}_{\text{T}_1} + \text{d}_1\,{}^{0}\mathbf{x}_{\text{B}^*} - \text{d}_2\,{}^{0}\mathbf{z}_{\text{H}^*} + \text{d}_3\,{}^{0}\mathbf{x}_{\text{H}^*} + \text{r}_{\text{F}}\,{}^{0}\mathbf{h} \,. \tag{C.5} $$

Here, r_{R}, r_{F}, d_1, d_2, and d_3 are bicycle geometries independent of the bicycle's configuration. The wheelbase w of the Whipple bicycle in its current configuration reads

$$ w = |\mathbf{r}_{\text{PQ}}| \,. \tag{C.6} $$

Setting the constraint that both wheels each touch the ground at one point leads to a scalar equation reading

$$ {}^{0}\mathbf{r}_{\text{PQ}} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\text{T}} = 0 \,. \tag{C.7} $$

If the bicycle's minimal coordinates are given, the only unknown in this equation is the pitch angle θ_B. Consequently, a calculation of it with Newton's method with the initial guess $\theta_B = 0$ [43] is possible.

A function for the pitch angular velocity $\dot{\theta}_B$ of the shape $\dot{\theta}_B(\theta_B, \varphi, \delta, \dot{\varphi}, \dot{\delta})$ can be gained by deriving Equation (C.7) with respect to time. Thus, also the pitch angular velocity $\dot{\theta}_B$ is known.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1007/s11044-026-10144-x.

Acknowledgements The computational results presented here have been achieved partly using the LEO HPC infrastructure of the University of Innsbruck.

Author contributions S.W.: formal analysis, data curation, investigation, methodology, software, validation, visualization, and writing of the original draft. P.M.: methodology, software, data curation, formal analysis, and conceptualization. A.S.: Formal analysis, conceptualization, and writing of original draft. J.G.: conceptualization, methodology, project administration, and supervision. All authors reviewed and approved the manuscript.

Funding information Open access funding provided by University of Innsbruck and Medical University of Innsbruck.

Data availability The data that support the findings of this study are available from the corresponding author, Johannes Gerstmayr, upon reasonable request.

Preprint An early stage version of this work has been uploaded as a preprint to arXiv, available at https://doi.org/10.48550/arXiv.2407.17156.

Declarations

Competing interests Non-financial interests: Johannes Gerstmayr and A. L. Schwab are members of the Editorial Advisory Board of Multibody System Dynamics.

References

1. Choi, H.-S., An, J., Han, S., Kim, J.-G., Jung, J.-Y., Choi, J., Orzechowski, G., Mikkola, A., Choi, J.H.: Data-driven simulation for general-purpose multibody dynamics using Deep Neural Networks. Multibody Syst. Dyn. **51**, 419–454 (2021). https://doi.org/10.1007/s11044-020-09772-8
2. Gerstmayr, J., Manzl, P., Pieber, M.: Multibody models generated from natural language. Multibody Syst. Dyn. **62**, 249–271 (2024). https://doi.org/10.1007/s11044-023-09962-0
3. Hashemi, A., Orzechowski, G., Mikkola, A., McPhee, J.: Multibody dynamics and control using machine learning. Multibody Syst. Dyn. **58**, 397–431 (2023). https://doi.org/10.1007/s11044-023-09884-x
4. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (2020). http://incompleteideas.net/book/the-book-2nd.html
5. Zai, A., Brown, B.: Deep Reinforcement Learning in Action. Manning Publications, Shelter Island (2020)

6. Sutton, R.S., Barto, A.G., Williams, R.J.: Reinforcement learning is direct adaptive optimal control. IEEE Control Syst. Mag. **12** (1992). https://doi.org/10.1109/37.126844
7. Hanakam, Y.: Querstabilisierung elektrisch unterstützter Fahrräder bei niedrigen Geschwindigkeiten. Ph.D. thesis, Universität Rostock (2023)
8. Meijaard, J.P., Papadopoulos, J.M., Ruina, A., Schwab, A.L.: Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review. Proc. Royal Soc. A **463**, 1955–1982 (2007). https://doi.org/10.1098/rspa.2007.1857
9. Kooijman, J.D.G., Meijaard, J.P., Papadopoulos, J.M., Ruina, A., Schwab, A.L.: A bicycle can be self-stable without gyroscopic or caster effects. Science **332**(6027), 339–342 (2011). https://doi.org/10.1126/science.1201959
10. Schwab, A.L., Meijaard, J.P.: A review on bicycle dynamics and rider control. Veh. Syst. Dyn. **51**, 1059–1090 (2013). https://doi.org/10.1080/00423114.2013.793365
11. Yin, S., Yamakita, M.: Passive velocity field control approach to bicycle robot path following. In: 2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 1654–1659 (2016). https://doi.org/10.1109/SICE.2016.7749208
12. Turnwald, A., Schäfer, M., Liu, S.: Passivity-based trajectory tracking control for an autonomous bicycle. In: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, pp. 2607–2612 (2018). https://doi.org/10.1109/IECON.2018.8591382
13. Shafiei, M.H., Emami, M.: Design of a robust path tracking controller for an unmanned bicycle with guaranteed stability of roll dynamics. Syst. Sci. Control Eng. **7**(1), 12–19 (2019). https://doi.org/10.1080/21642583.2018.1555062
14. Seekhao, P., Tungpimolrut, K., Parnichkun, M.: Development and control of a bicycle robot based on steering and pendulum balancing. Mechatronics **69** (2020). https://doi.org/10.1016/j.mechatronics.2020.102386
15. Persson, N., Ekström, M.C., Ekström, M., Papadopoulos, A.V.: Trajectory tracking and stabilisation of a riderless bicycle. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp. 1859–1866 (2021). https://doi.org/10.1109/ITSC48978.2021.9564958
16. He, K., Deng, Y., Wang, G., Sun, X., Sun, Y., Chen, Z.: Learning-based trajectory tracking and balance control for bicycle robots with a pendulum: a Gaussian process approach. IEEE/ASME Trans. Mechatron. **27**(2), 634–644 (2022). https://doi.org/10.1109/TMECH.2022.3140885
17. Randlov, J., Alstrøm, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: Proceedings of the Fifteenth International Conference on Machine Learning. ICML'98, pp. 463–471. Association for Computing Machinery (1998)
18. Le, T.P., Chung, T.: Controlling bicycle using deep deterministic policy gradient algorithm. In: 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 413–417 (2017). https://doi.org/10.1109/URAI.2017.7992765
19. Le, T.P., Choi, S., Nguyen, Q.D., Layek, M.A., Lee, S., Chung, T.: Toward self-driving bicycles using state-of-the-art deep reinforcement learning algorithms. Symmetry **11**(2) (2019). https://doi.org/10.3390/sym11020290
20. Cook, M.: It takes two neurons to ride a bicycle. In: Advances in Neural Information Processing Systems 17 (NIPS 2014) (demo) (2004). https://paradise.caltech.edu/cook/papers/TwoNeurons.pdf
21. Zhu, X., Zheng, X., Zhang, Q., Chen, Z., Liu, Y., Liang, B.: Natural residual reinforcement learning for bicycle robot control. In: 2021 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1201–1206 (2021). https://doi.org/10.1109/ICMA52036.2021.9512587
22. Zhu, X., Deng, Y., Zheng, X., Zheng, Q., Chen, Z., Liang, B., Liu, Y.: Online series-parallel reinforcement-learning-based balancing control for reaction wheel bicycle robots on a curved pavement. IEEE Access **11**, 66756–66766 (2023). https://doi.org/10.1109/ACCESS.2023.3268524
23. Huo, B., Yu, L., Liu, Y., Chen, Z.: Hierarchical residual reinforcement learning based path tracking control method for unmanned bicycle. Robot. Auton. Syst. **190**, 104996 (2025). https://doi.org/10.1016/j.robot.2025.104996
24. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Stanley: the robot that won the DARPA grand challenge. J. Field Robot. **23**(9), 661–692 (2006). https://doi.org/10.1002/rob.20147
25. Whipple, F.J.: The stability of the motion of a bicycle. Q. J. Pure Appl. Math. **30**(120), 312–348 (1899)
26. Xiong, J., Wang, N., Liu, C.: Stability analysis for the Whipple bicycle dynamics. Multibody Syst. Dyn. **48**, 311–335 (2020). https://doi.org/10.1007/s11044-019-09707-y
27. Schwab, A.L., Kooijman, J.D.G., Meijaard, J.P.: Some recent developments in bicycle dynamics and control. In: Fourth European Conference on Structural Control (4ECSC) (2008). http://bicycle.tudelft.nl/schwab/Publications/SchwabKooijmanMeijaard2008.pdf

28. Zwölfer, A., Gerstmayr, J.: A concise nodal-based derivation of the floating frame of reference formulation for displacement-based solid finite elements. Multibody Syst. Dyn. **49**, 291–313 (2020). https://doi.org/10.1007/s11044-019-09716-x
29. Bauchau, O.A., Rodriguez, J.: Modeling of joints with clearance in flexible multibody systems. Int. J. Solids Struct. **39**(1), 41–63 (2002). https://doi.org/10.1016/S0020-7683(01)00186-X
30. Achiam, J.: Spinning up in Deep Reinforcement Learning (2018)
31. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (2018). arXiv preprint. https://doi.org/10.48550/arXiv.1801.01290
32. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-Baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**(268), 1–8 (2021)
33. Gerstmayr, J.: Exudyn – a C++ based Python package for flexible multibody systems. Multibody Syst. Dyn. **60**, 533–561 (2023). https://doi.org/10.1007/s11044-023-09937-1
34. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016). arXiv preprint. https://doi.org/10.48550/arXiv.1606.01540
35. Astrom, K.J., Klein, R.E., Lennartsson, A.: Bicycle dynamics and control: adapted bicycles for education and research. IEEE Control Syst. Mag. **25**(4), 26–47 (2005). https://doi.org/10.1109/MCS.2005.1499389
36. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ICML'09, pp. 41–48 (2009). https://doi.org/10.1145/1553374.1553380
37. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 1928–1937 (2016). https://doi.org/10.48550/arXiv.1602.01783
38. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (2017). arXiv preprint. https://doi.org/10.48550/arXiv.1707.06347
39. Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30 (NIPS 2017), vol. 30 (2017). https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions
40. Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 737–744 (2020). https://doi.org/10.1109/SSCI47803.2020.9308468
41. Henderson, D.M.: Euler angles, quaternions, and transformation matrices for space shuttle analysis. Techreport, NASA (1977). https://ntrs.nasa.gov/citations/19770019231
42. Psiaki, M.L.: Bicycle Stability: a Mathematical and Numerical Analysis. Bachelor thesis, Princeton University (1979). http://ruina.tam.cornell.edu/research/topics/bicycle_mechanics/Psiaki_Princeton_thesis.pdf
43. Peterson, D., Hubbard, M.: Analysis of the holonomic constraint in the Whipple bicycle model. In: Estivalet, M., Brisson, P. (eds.) The Engineering of Sport 7, vol. 2 (2008). https://doi.org/10.1007/978-2-287-99056-4_75