Learning a Low-Dimensional Representation of a Safe Region for Safe Reinforcement Learning on Dynamical Systems

Zhehua Zhou, *Graduate Student Member, IEEE*, Ozgur S. Oguz, *Member, IEEE*, Marion Leibold, *Member, IEEE*, and Martin Buss, *Fellow, IEEE*

***Abstract*— For the safe application of reinforcement learning algorithms to high-dimensional nonlinear dynamical systems, a simplified system model is used to formulate a safe reinforcement learning (SRL) framework. Based on the simplified system model, a low-dimensional representation of the safe region is identified and used to provide safety estimates for learning algorithms. However, finding a satisfying simplified system model for complex dynamical systems usually requires a considerable amount of effort. To overcome this limitation, we propose a general data-driven approach that is able to efficiently learn a low-dimensional representation of the safe region. By employing an online adaptation method, the low-dimensional representation is updated using the feedback data to obtain more accurate safety estimates. The performance of the proposed approach for identifying the low-dimensional representation of the safe region is illustrated using the example of a quadcopter. The results demonstrate a more reliable and representative low-dimensional representation of the safe region compared with previous works, which extends the applicability of the SRL framework.**

***Index Terms*— Data-driven model order reduction, deep learning in robotics and automation, learning and adaptive systems, safe reinforcement learning (SRL).**

I. Introduction

RECENT studies of applying reinforcement learning or deep reinforcement learning algorithms to complex, i.e., highly nonlinear and high-dimensional, dynamical systems have demonstrated attractive achievements in various control tasks, e.g., humanoid control [1] and robotic manipulator control [2]. However, although the results display the potential of utilizing reinforcement learning algorithms as a substitute for traditional controller design techniques, most of them are still only presented in simulations [3]. One major impediment against implementing reinforcement learning algorithms on real-world dynamical systems is that, due to the random exploration mechanism, the intermediate policy may lead to dangerous behaviors of the system. As a result, both the system itself and the environment may be damaged during learning. In order to apply state-of-the-art reinforcement learning algorithms to real-world control systems, one central problem to address is how to introduce a reliable safety guarantee into the learning process.

Manuscript received 16 October 2020; revised 7 July 2021; accepted 14 August 2021. Date of publication 3 September 2021; date of current version 3 May 2023. *(Corresponding author: Zhehua Zhou.)*

Zhehua Zhou, Marion Leibold, and Martin Buss are with the Chair of Automatic Control Engineering, Technical University of Munich, 80290 Munich, Germany (e-mail: zhehua.zhou@tum.de; marion.leibold@tum.de; mb@tum.de).

Ozgur S. Oguz is with the Max Planck Institute for Intelligent Systems, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: ozgur.oguz@ipvs.uni-stuttgart.de).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3106818.

Digital Object Identifier 10.1109/TNNLS.2021.3106818

A. Related Work

Safe reinforcement learning (SRL) aims to find an optimal control policy by way of reinforcement learning while ensuring that certain safety conditions are not violated during the learning process. Although the exact definition of safety in SRL varies in different learning tasks, for instance, collision avoidance in autonomous vehicles or crash prevention when controlling a quadcopter, we generally consider the safety condition as neither the system itself nor the environment will be damaged.

SRL in dynamical systems with continuous action space has been a topic of research for over a decade [4]. Most previous studies employed a manual control mechanism to ensure the safety of the controlled system. For instance, in [5], an experienced human pilot takes over the control of the helicopter if the learning algorithm places the system in a risky state. However, such an approach requires a considerable amount of resources to monitor the entire learning process. Hence, in most cases, it is not applicable to complex learning tasks. Another possibility of safely implementing reinforcement learning algorithms on real-world dynamical systems is by transfer learning [6]. First, a satisfying initial policy is trained in simulation and then transferred to the real-world dynamical system. In essence, this minimizes the required number of learning iterations for obtaining the final policy, and thus, reduces the risk of encountering dangerous intermediate policy [7]. However, since the mismatch between simulation and reality is not considered in transfer learning, no reliable safety guarantee is obtained [8].

In recent studies, SRL in model-free scenarios is usually achieved by solving a constraint satisfaction problem. For example, constrained policy optimization [9] introduces a constraint to the learning process to the effect that the expected return of cost functions should not exceed certain predefined limits. Alternatively, including an additional risk term in the reward function, such as risk-sensitive reinforcement learning [10], can also increase the safety of reinforcement learning algorithms. However, as no system model is directly considered in these approaches, there is still a high possibility that safety conditions are violated, especially in the early learning phase.

When at least an approximated system model is available, a more promising SRL can be realized by combining control-theoretic concepts with reinforcement learning approaches. For example, in [11] and [12], Lyapunov functions are employed to compute a subregion of the state space where safety conditions will never be violated. The system is then limited to this subregion during the learning process. However, finding suitable candidates for Lyapunov functions is challenging if the system dynamics contain uncertainties or are highly nonlinear.

For uncertain dynamical systems, methods based on learning a model of unknown system dynamics [13] or of environmental constraints [14] are proposed to ensure safety during the learning. For instance, by predicting the system behavior in the worst case, robust model predictive control [15] is able to provide safety and stability guarantees to reinforcement learning algorithms if the error in the learned model is bounded. Besides, [16] introduces an action governor to correct the applied action when the system is predicted to be unsafe. However, limited by computational efficiency, these approaches with deterministic safety estimates, i.e., the prediction about the safety of a system state is either safe or unsafe, are usually only applicable to linear systems. Moreover, the accuracy of the learned model also strongly affects the performance of these approaches.

To relax the demands placed on the system model and extend the SRL to nonlinear systems, instead of deterministic safety estimates, recent studies employ probabilistic safety estimates, in which safety predictions are represented as probabilities [17]. In [18], for example, modeling uncertainties are approximated by Gaussian process models [19], and a probabilistic safe region is computed by reachability analysis [20]. Similarly, Gaussian process models are used in [21] and [22] to model unknown system dynamics. A safe region is then obtained from the probabilistic estimate of the region of attraction (ROA) of a safe equilibrium state. The key component of these studies is a forward invariant safe region, such that the learning algorithm has the flexibility to execute desired actions within the safe region. Safety is ensured by switching to a safety controller whenever the system approaches the boundary of the safe region. However, the safe region is computed either by solving a partial differential equation in [18] or sampling in [22], both of which suffer from the curse of dimensionality. Moreover, modeling an unknown dynamics or disturbance with Gaussian process models also poses challenges when the system is highly nonlinear and high dimensional since both making adequate assumptions about the distribution of dynamics and acquiring a sufficient amount of data are difficult. Therefore, although approaches, such as [18] and [22], enable promising results with low-dimensional dynamical systems,[1] they are not directly applicable to complex dynamical systems [23].

Often the motivation for using reinforcement learning algorithms for controller design is to overcome the difficulty of applying model-based controller design approaches to highly nonlinear, high-dimensional, and uncertain dynamic system models [24], [25]. In particular, it is challenging to compute a safe region for a complex dynamical system. For this reason, [26] introduces an SRL framework that utilizes a supervisory control strategy based on finding a simplified system by means of physically inspired model order reduction [27]. A simplified safe region is constructed from the simplified system, which functions as an approximation for the safe region of the full dynamics. Such a low-dimensional representation of the safe region, which is usually 2-D or 3-D, at least provides safety estimates for the original system states, and it can be updated online during the learning process. To account for the uncertainty in making safety decisions for the complex dynamics based on a rough low-dimensional reduction, the safety estimate is represented in a probabilistic form. Then, in accordance with the derived safety estimate, a supervisor is employed to switch the actually applied control action between the learning algorithm and a corrective controller to keep the system safe. However, implementing physically inspired model order reduction usually requires a thorough understanding of the system dynamics. Moreover, multiple performance tests are required before a satisfying simplified system can be found.

B. Contribution

In this paper, we consider the same supervisory control strategy as used in [26] to construct a general SRL framework that is applicable to complex dynamical systems. However, to overcome the limitations of physically inspired model order reduction, we propose a novel data-driven approach to identify the supervisor, i.e., the low-dimensional representation of the safe region. Inspired by transfer learning [28], we assume that an approximated system model of the complex dynamical system is available. Even though, inevitably, the approximated model displays discrepancies compared with the real system behavior, an initial estimate of safety can usually be obtained by simulating the approximated model. For example, while the dynamics of a real-world humanoid cannot be known perfectly, an approximated humanoid model can be constructed in simulation for making predictions. Hence, by simulating the system, we obtain training data that represents the safety of various original system states. However, as the state space is high dimensional, it is infeasible to acquire a sufficient amount of training data to directly learn the safe region of the original system. To solve this problem, a data-driven approach that computes probabilistic similarities between each training

[1]In this article, we consider dynamical systems with dimensions higher than six as high dimensional, as in such cases it is computationally difficult to implement traditional methods, such as reachability analysis or sum-of-squares programming, in identifying the safe region.

data is proposed to first learn a low-dimensional representative safety feature of the complex dynamical system. Then, based on the learned feature, a low-dimensional representation of the safe region is identified, which is used as the starting point to SRL in the real system.

Due to the inevitable simulation-to-reality gap, the initial low-dimensional representation of the safe region learned from training data displays discrepancies compared to the real system behavior. To compensate for this mismatch, we also propose an efficient online adaptation method to update the low-dimensional representation of the safe region. During the learning process, we receive feedback data about the actual safe region of the real system. These feedback data are not only used to generate new safety estimates, but they also allow us to adjust our confidence in the reliability of the safety estimates obtained from training data. The proposed online adaptation method then updates the low-dimensional representation of the safe region by simultaneously considering the safety estimates derived from training and feedback data.

The contributions of this study are summarized as follows.

1) We propose a novel data-driven approach that is capable of systematically identifying a low-dimensional representation of the safe region. In contrast to physically inspired model order reduction, the proposed approach does not require a thorough understanding of system dynamics. Moreover, it is applicable to a wide range of dynamical systems, as long as an approximated system model is available.
2) We introduce a new online adaptation method for updating the low-dimensional representation of the safe region according to the observed real system behavior. By fully utilizing the information contained in the feedback data, the update is performed efficiently, while a reasonable amount of feedback data enables an accurate low-dimensional representation of the safe region to be acquired.
3) Since the proposed approach results in a reliable and representative low-dimensional representation of the safe region, the applicability of the SRL framework is increased.

The remainder of this article is organized as follows: a brief introduction to the SRL framework is given in Section II. Thereafter, we present an overview of our approach in Section III. In Section IV, we propose a data-driven method to derive a low-dimensional representation of the safe region. This is followed by the online adaptation method in Section V, which is used to update the low-dimensional representation. An example is presented in Section VI to demonstrate the performance of the proposed approach. In Section VII, we discuss several properties of the approach, and Section VIII concludes the article. A table of nomenclatures is included in the Supplementary Material.

II. Safe Reinforcement Learning Framework

In this article, we consider SRL in order to optimize a learning-based policy with respect to a predefined reward function, while ensuring that the system state remains in a safe region of the state space. In this section, we outline a general SRL framework for dynamical systems, see also [26]. The SRL framework first identifies a safe state-space region as the safe region. Then, the learning-based policy has the flexibility to execute desired actions within the safe region. Once the system state is about to leave the safe region, a corrective controller is applied to drive the system back to a safe state.

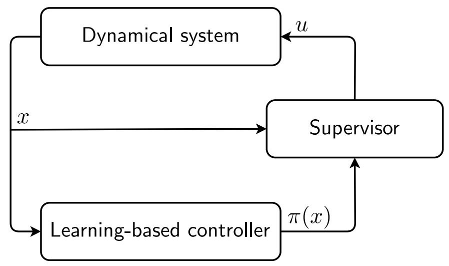


Fig. 1. SRL framework with a supervisor which decides on the actual applied actions.

A. System Model and Safe Region

A nonlinear control-affine dynamical system is given by

$$\dot{x} = f(x) + g(x)u \tag{1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the n-dimensional system state within a connected set $\mathcal{X}$, and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the m-dimensional control input to the system. With a given control policy $u = K(x)$, the closed-loop system dynamics is denoted as

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

If a system state x satisfies $f_K(x) = 0$, then it is an equilibrium point. Any equilibrium point can be shifted to the origin by a state transform. Therefore, this article only uses the origin to formulate the safe region.

Assumption 1: The origin is a safe state and a locally asymptotically stable equilibrium point under the control policy $K(x)$.

Based on Assumption 1, the ROA of the origin is defined as

$$\mathcal{R} = \left\{x_0 \in \mathcal{X} \mid \lim_{t\to\infty} \Phi(t; x_0) = 0\right\} \tag{3}$$

where $\Phi(t; x_0)$ is the system trajectory of (2) that starts at the initial state x_0 when time $t = 0$. The ROA $\mathcal{R}$ is the set of initial states that can be driven back to a safe state, i.e., the origin, under the control policy $K(x)$. Therefore, in this article, we define the safe region of the SRL framework as follows.

Definition 1: A safe region $\mathcal{S}$ is a closed positive invariant subset of the ROA $\mathcal{R}$ containing the origin. We consider the system state x as safe if it is in the safe region $\mathcal{S}$.

B. SRL Framework

To realize SRL, we keep the system state within the safe region during the learning process. This is achieved by an SRL framework that adapts a switching supervisory control strategy, where the given controller $K(x)$ acts as corrective control and $\pi(x)$ is the learning-based policy that is used while the system state is in the safe region (see Fig. 1). A supervisor determines the actual applied actions as

$$u = \begin{cases} \pi(x), & \text{if } t < t^{\text{safe}} \\ K(x), & \text{else} \end{cases} \tag{4}$$

where t^{safe} is the first time point at which the system state x is on the boundary of the safe region $\mathcal{S}$.

For each learning iteration, the system starts inside the safe region $\mathcal{S}$ for time $t = 0$. The learning algorithm then updates and executes the learning-based policy $\pi(x)$. Since the safe region $\mathcal{S}$ is a closed set and the trajectory is continuous, the system state can only leave the safe region $\mathcal{S}$ by crossing the boundary. Hence, once the system state x is on the boundary of the safe region $\mathcal{S}$, this learning iteration is terminated at time $t = t^{\text{safe}}$ and the corrective controller $K(x)$ is activated. For the remaining time of this learning iteration, the corrective controller $K(x)$ attempts to bring the system back to the origin to maintain safety. After this safety recovery, the learning environment is reset and the next learning iteration starts at time $t = 0$.

Remark 1: In this article, we only consider the safe region obtained from the ROA $\mathcal{R}$, where stability is used as the safety criterion. If more safety criteria should be taken into consideration, such as collision avoidance represented as state constraints, the safe region can be constructed using other control-theoretical concepts, e.g., control barrier functions [29] or invariance functions [30]. The definition of the safe region does not affect the use of the SRL framework and the proposed approach, as long as the safe region is a closed and control invariant set under a given corrective controller.

C. SRL Framework for Complex Dynamical Systems

The aforementioned SRL framework is not directly applicable to complex dynamical systems, as in such cases, calculating the safe region $\mathcal{S}$ is computationally infeasible [31]. An SRL framework based on estimating safety with a low-dimensional representation of the safe region is introduced to overcome this problem [26].

Each original system state x is mapped to a low-dimensional safety feature, represented as a simplified state $y \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$, $n_y \ll n$, through a state mapping $y = \Psi(x)$. The state mapping is chosen such that safe and unsafe states are separated in the simplified state space $\mathcal{Y}$. Nevertheless, due to the order reduction, multiple original system states that have different safety properties can map to the same simplified state. Hence, the safety of the original system state x is estimated by the safety of its corresponding simplified state y in a probabilistic form as

$$p(x \in \mathcal{S}) = \Gamma(y)|_{y=\Psi(x)} \sim [0, 1] \tag{5}$$

where $\Gamma(y)$ is a function defined over the simplified state space $\mathcal{Y}$ and is referred to as the *safety assessment function* (SAF) in this article. Not only does the SAF $\Gamma(y)$ encode information relating to the safety of the simplified state y, it also includes the uncertainty involved in making predictions for a high-dimensional state by using a low-dimensional reduction. In Section IV, we demonstrate how to efficiently identify the state mapping $y = \Psi(x)$ and the SAF $\Gamma(y)$ using a data-driven method.

For a given SAF $\Gamma(y)$, the probability $p(x \in \mathcal{S})$ depends only on the simplified state y. Therefore, by introducing a predefined probability threshold p_t, we obtain a low-dimensional representation of the safe region, denoted as $\mathcal{S}_y$, in the simplified state space $\mathcal{Y}$

$$\mathcal{S}_y = \{y \in \mathcal{Y} \mid \Gamma(y) > p_t\} \tag{6}$$

which works as an approximation of the high-dimensional safe region $\mathcal{S}$. The supervisor (4) is, thus, modified to

$$u = \begin{cases} \pi(x), & \text{if } t < t^{\text{safe}'} \\ K(x), & \text{else} \end{cases} \tag{7}$$

where $t^{\text{safe}'}$ denotes the first time point at which the probability $p(x \in \mathcal{S})$ is not larger than the threshold p_t, i.e., $p(x \in \mathcal{S}) = \Gamma(y) \leq p_t$. More details of this SRL framework are given in [26].

III. Overview of the Approach

The essential factor when applying the SRL framework to complex dynamical systems is finding a reliable low-dimensional representation of the safe region $\mathcal{S}_y$. In order to overcome the limitations of physically inspired model order reduction, we propose a novel data-driven approach to identify the low-dimensional representation of the safe region $\mathcal{S}_y$, together with a new online adaptation method to efficiently update the learned low-dimensional representation.

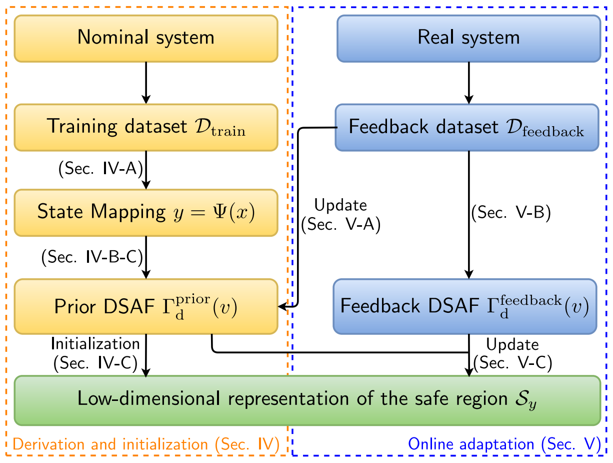
We consider a scenario in which the complex dynamical system, referred to as the real system, has partially unknown dynamics. However, we assume that a nominal approximated system model is available and can be used to roughly predict the real system behavior. The nominal system model is assumed to be represented by (1). The real-system model is then given as

$$\dot{x} = f(x) + g(x)u + d(x) \tag{8}$$

where $d(x)$ is the unknown, unmodeled part of the system dynamics. For brevity, we refer to the nominal and the real systems as *simulation* and *reality*, respectively.

Due to the highly nonlinear and high-dimensional dynamics, the direct calculation of the safe region is computationally infeasible for both the nominal and the real systems. Besides, although the real system provides exact safety information, in general, it is expensive to collect data directly on the real system. In contrast, simulating the nominal system is usually efficient and allows a sufficient amount of data to be obtained for finding a low-dimensional safety representation. However, due to the unknown term $d(x)$, such data are inaccurate and have to be modified to account for the real-system behavior.

Based on these facts, to construct a reliable low-dimensional representation of the safe region $\mathcal{S}_y$ for the real system, we propose the approach outlined in Fig. 2 (a complete workflow is given in the Supplementary Material). It consists of two parts that solve the following two problems, respectively.



Nominal system
Training dataset $\mathcal{D}_{\text{train}}$
(Sec. IV-A)
State Mapping $y = \Psi(x)$
(Sec. IV-B-C)
Prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$
Initialization (Sec. IV-C)
Real system
Feedback dataset $\mathcal{D}_{\text{feedback}}$
Update (Sec. V-A)
(Sec. V-B)
Feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$
Update (Sec. V-C)
Low-dimensional representation of the safe region $\mathcal{S}_y$
Derivation and initialization (Sec. IV)
Online adaptation (Sec. V)

Fig. 2. Overview of the proposed approach. The low-dimensional representation $\mathcal{S}_y$ is initialized using the training dataset $\mathcal{D}_{\text{train}}$ obtained from the nominal system. Once we collect the feedback dataset $\mathcal{D}_{\text{feedback}}$ on the real system, the low-dimensional representation $\mathcal{S}_y$ is updated using the proposed online adaptation method.

1) How to derive and initialize the low-dimensional representation of the safe region $\mathcal{S}_y$ by using the nominal system model.
2) How to update the low-dimensional representation of the safe region $\mathcal{S}_y$ online with the observed real-system behavior.

A. Part 1: Derivation and Initialization

Since no information about uncertainty $d(x)$ is available prior to the learning process, the corrective controller $K(x)$ is designed for the nominal system model (1). Although the safe region of the nominal system is unknown, its simulation is possible and delivers a dataset as follows.

Definition 2: The training dataset of k_t training data is given as

$$\mathcal{D}_{\text{train}} = \left\{ D_{\text{train}}^1, D_{\text{train}}^2, \ldots, D_{\text{train}}^{k_t} \right\}. \tag{9}$$

It contains the simulation results that state whether the safety recovery is successful or not for different system states x under the corrective controller $K(x)$. The ith training data consists of three elements

$$D_{\text{train}}^i = \left\{ x_{\text{sim}}^i, s_{\text{sim}}\left(x_{\text{sim}}^i\right), \Phi_{\text{sim}}\left(t; x_{\text{sim}}^i\right) \right\}. \tag{10}$$

x_{sim}^i is the initial system state in which the corrective controller $K(x)$ is activated. $s_{\text{sim}}(x_{\text{sim}}^i)$ is the safety label that represents the result of safety recovery for the state x_{sim}^i. We denote $s_{\text{sim}}(x_{\text{sim}}^i) = 1$ if the system state x_{sim}^i is safe under the corrective controller $K(x)$, and $s_{\text{sim}}(x_{\text{sim}}^i) = 0$ if it is not. $\Phi_{\text{sim}}(t; x_{\text{sim}}^i)$ is the corresponding system trajectory of the safety recovery that starts at x_{sim}^i when time $t = 0$. The subscript sim indicates that the data is collected by using the nominal system model.

The low-dimensional representation of the safe region $\mathcal{S}_y$ is, thus, derived and initialized by using the training dataset $\mathcal{D}_{\text{train}}$. To do this, we first identify the state mapping $y = \Psi(x)$ using a data-driven method that computes the probabilistic similarity between each training data (Section IV-A). Then, to facilitate an efficient computation, we discretize the simplified state space $\mathcal{Y}$ into grid cells and assign an index vector $v \in \mathbb{Z}_+^{n_y}$ to each grid cell. By assuming that the SAF $\Gamma(y)$ is constant in each grid cell, we, thus, obtain a *discretized SAF (DSAF)* $\Gamma_d(v)$. A discretized low-dimensional representation of the safe region $\mathcal{S}_y$ is then given by applying the probability threshold p_t on the DSAF $\Gamma_d(v)$ (Section IV-B). To enable the SRL framework on the real system, we also calculate an initial estimate of the DSAF $\Gamma_d(v)$, denoted as the prior DSAF $\Gamma_d^{\text{prior}}(v)$, from the training dataset $\mathcal{D}_{\text{train}}$. It is then used to initialize the low-dimensional representation of the safe region $\mathcal{S}_y$ (Section IV-C). Further details of Part 1 are given in Section IV.

B. Part 2: Online Adaptation

Due to the unknown part of the system dynamics $d(x)$, there is inevitably a mismatch between simulation and reality. In order to compensate for this mismatch, we update the low-dimensional representation $\mathcal{S}_y$ by accounting for the real-system behavior.

Each time the corrective controller $K(x)$ is activated during learning, we observe feedback data about the real safe region. The set of feedback data is defined as follows.

Definition 3: The feedback dataset of k_f feedback data is given as

$$\mathcal{D}_{\text{feedback}} = \left\{ D_{\text{feedback}}^1, D_{\text{feedback}}^2, \ldots, D_{\text{feedback}}^{k_f} \right\}. \tag{11}$$

It contains the results of safety recovery from implementing the corrective controller $K(x)$ on the real system. The ith feedback data are

$$D_{\text{feedback}}^i = \left\{ x_{\text{real}}^i, s_{\text{real}}\left(x_{\text{real}}^i\right), \Phi_{\text{real}}\left(t; x_{\text{real}}^i\right) \right\}. \tag{12}$$

While x_{real}^i, $s_{\text{real}}(x_{\text{real}}^i)$, and $\Phi_{\text{real}}(t; x_{\text{real}}^i)$ have the same meaning as in Definition 2, the subscript real indicates here that the data are collected on the real system.

Since collecting data on the real system, e.g., real-world robots, is usually expensive and time-consuming, in most cases, the feedback dataset $\mathcal{D}_{\text{feedback}}$ has a limited size. Therefore, the low-dimensional representation of the safe region $\mathcal{S}_y$ needs to be updated in a data-efficient manner. To achieve this, we propose an online adaptation method, as given in Section V. It comprises three steps: first, we modify the prior DSAF $\Gamma_d^{\text{prior}}(v)$ by changing our confidence in its reliability using the feedback dataset $\mathcal{D}_{\text{feedback}}$ (Section V-A). Second, to fully utilize the valuable information contained in the feedback dataset $\mathcal{D}_{\text{feedback}}$, we generate another feedback DSAF $\Gamma_d^{\text{feedback}}(v)$ (Section V-B). Third, the two DSAFs are fused to obtain a more accurate DSAF $\Gamma_d(v)$, which is then used to update the low-dimensional representation $\mathcal{S}_y$ (Section V-C).

IV. Learning a Low-Dimensional Representation of the Safe Region

To derive the low-dimensional representation of the safe region $\mathcal{S}_y$, two components have to be determined: the state mapping $y = \Psi(x)$, which gives the low-dimensional safety feature, and the SAF $\Gamma(y)$, which predicts the safety of original system states. In this section, we present a data-driven method for identifying the low-dimensional representation of the safe region $\mathcal{S}_y$. It utilizes a technique called t-distributed stochastic neighbor embedding (t-SNE) [32], which was originally proposed for visualizing high-dimensional data.

A. Identifying the State Mapping With t-SNE

To identify the state mapping $y = \Psi(x)$, we first find the realization of the low-dimensional safety feature, i.e., the values of simplified states $y^1, \ldots, y^{k_t}$, that best corresponds with the training dataset $\mathcal{D}_{\text{train}}$ by revising t-SNE. Through measuring the similarity between each high-dimensional data point, t-SNE defines a 2-D or 3-D data point such that similar high-dimensional data points are represented by nearby low-dimensional data points with high probability. It uses Euclidean distance between each pair of high-dimensional data points as the metric for measuring similarity. However, since our purpose is to construct the low-dimensional representation of the safe region $\mathcal{S}_y$, we are more interested in safety, rather than just distance. Accordingly, we propose a new metric that considers similarity and safety at the same time.

The general motivation for determining the simplified state y is that the safe and unsafe original system states x should be separated in the simplified state space $\mathcal{Y}$. Since, in this article, the safe region is defined with respect to the ROA, the trajectories of safe initial states will converge to the origin, while unsafe initial states will have divergent trajectories. Hence, if two original system states x have similar trajectories under the corrective controller $K(x)$, then ideally they should also have nearby corresponding simplified states y (see Fig. 3). Based on this, we first calculate the pairwise trajectory distance ω_{ij} between the ith and jth training data, using dynamic time warping (DTW) as

$$\omega_{ij} = \text{dtw}\left(\Phi_{\text{sim}}\left(t; x_{\text{sim}}^i\right), \Phi_{\text{sim}}\left(t; x_{\text{sim}}^j\right)\right) \tag{13}$$

where dtw$(\cdot)$ represents the DTW measurement. We, thus, have $\omega_{ij} = 0$ if $i = j$, and the more similar the trajectories are, the smaller the value of ω_{ij} is.

Remark 2: Besides DTW, other trajectory distance measures, e.g., Fréchet distance, can also be used in (13). Changing the distance metric does not affect the applicability of the proposed approach. However, DTW turns out to be a more suitable metric for trajectories of the dynamical systems we investigated.

While, in general, the trajectory distance ω_{ij} reflects the probability that the original system states x_{sim}^i and x_{sim}^j have the same safety property, it is still possible that safe and unsafe states have similar trajectories. To obtain a better low-dimensional safety feature, we, thus, modify the trajectory distance ω_{ij} in relation to the safety label $s_{\text{sim}}(x_{\text{sim}})$ and compute the distance Ω_{ij} between the ith and jth training data as

$$\Omega_{ij} = \begin{cases} \dfrac{\omega_{ij}}{\omega_{\max}} + \delta, & \text{if } s_{\text{sim}}\left(x_{\text{sim}}^i\right) \neq s_{\text{sim}}\left(x_{\text{sim}}^j\right) \\[2ex] \dfrac{\omega_{ij}}{\omega_{\max}}, & \text{if } s_{\text{sim}}\left(x_{\text{sim}}^i\right) = s_{\text{sim}}\left(x_{\text{sim}}^j\right) \end{cases} \tag{14}$$

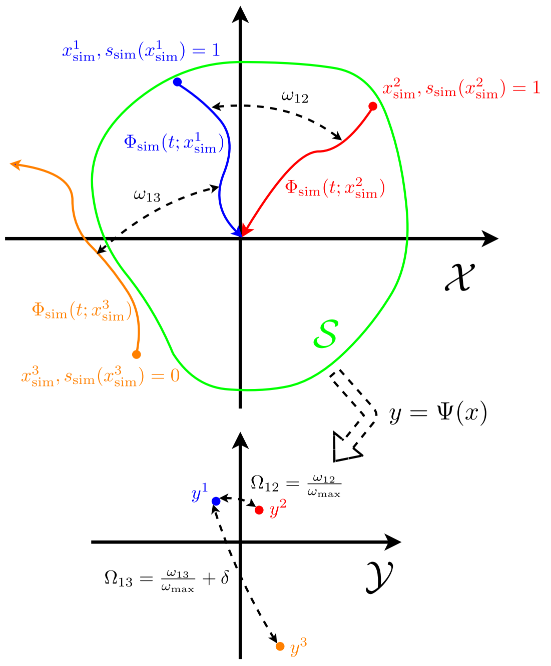

$x_{\text{sim}}^1, s_{\text{sim}}(x_{\text{sim}}^1) = 1$
ω_{12}
$x_{\text{sim}}^2, s_{\text{sim}}(x_{\text{sim}}^2) = 1$
$\Phi_{\text{sim}}(t; x_{\text{sim}}^1)$
ω_{13}
$\Phi_{\text{sim}}(t; x_{\text{sim}}^2)$
$\mathcal{X}$
$\Phi_{\text{sim}}(t; x_{\text{sim}}^3)$
$\mathcal{S}$
$x_{\text{sim}}^3, s_{\text{sim}}(x_{\text{sim}}^3) = 0$
$y = \Psi(x)$
y^1
$\Omega_{12} = \frac{\omega_{12}}{\omega_{\max}}$
y^2
$\Omega_{13} = \frac{\omega_{13}}{\omega_{\max}} + \delta$
$\mathcal{Y}$
y^3

Fig. 3. Distances Ω_{12} and Ω_{13} are computed for three training data D_{train}^1, D_{train}^2, and D_{train}^3 using the trajectory distances ω_{12} and ω_{13}, and the safety labels $s_{\text{sim}}(x_{\text{sim}}^1)$, $s_{\text{sim}}(x_{\text{sim}}^2)$, and $s_{\text{sim}}(x_{\text{sim}}^3)$. Based on these distances, t-SNE calculates the values of corresponding simplified states y, where similar and dissimilar training data are modeled by nearby and distant simplified states, respectively.

where δ is a constant and $\omega_{\max} = \max_{i,j} \omega_{ij}$ is the maximum trajectory distance within the training dataset $\mathcal{D}_{\text{train}}$. The distance Ω_{ij} is then used as the new metric for t-SNE to measure the similarities between different training data.

In our experiments, we find that a small value of δ is sufficient for providing a satisfying result of t-SNE (in this article, for example, we use $\delta = 0.01$). A large value of δ, in contrast, may lead to the information contained in trajectories being ignored, which can reduce the representation power of the learned simplified states y. A sensitivity analysis of the parameter δ is provided in the Supplementary Material.

After computing the distance Ω_{ij} between each pair of training data, we apply t-SNE on the training dataset $\mathcal{D}_{\text{train}}$ to derive a realization of the low-dimensional safety feature. To do this, we modify the conditional probability $p_{j|i}$ of t-SNE [32] using the distance Ω_{ij} as

$$p_{j|i} = \frac{\exp\left(-\Omega_{ij}^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\Omega_{ik}^2 / 2\sigma_i^2\right)} \tag{15}$$

where σ_i is the variance of the Gaussian distribution that is centered on the state x_{sim}^i. The remaining computations are the same as in t-SNE. Since this part makes no contribution, the main steps involved in performing t-SNE are given only in the Supplementary Material. More details are available in [32].

Using t-SNE, we obtain the values of simplified states $y^1, \ldots, y^{k_t}$ that correspond to the training dataset $\mathcal{D}_{\text{train}}$ as an initial realization of the low-dimensional safety feature. Such a realization models similar training data with nearby simplified states, e.g., y^1 and y^2 in Fig. 3, and dissimilar

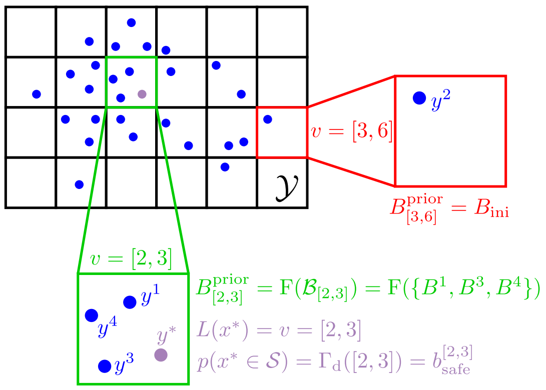

y^2
$\upsilon = [3,6]$
$\mathcal{Y}$
$B^{\text{prior}}_{[3,6]} = B_{\text{ini}}$
$\upsilon = [2,3]$
y^1
y^4
y^*
y^3
$B^{\text{prior}}_{[2,3]} = \text{F}(\mathcal{B}_{[2,3]}) = \text{F}(\{B^1, B^3, B^4\})$
$L(x^*) = \upsilon = [2,3]$
$p(x^* \in \mathcal{S}) = \Gamma_{\text{d}}([2,3]) = b^{[2,3]}_{\text{safe}}$

Fig. 4. Simplified state space $\mathcal{Y}$ is discretized into grid cells. The location of each grid cell is indicated by the index vector υ. The safety of a new original system state, e.g., x^*, is estimated by way of the corresponding belief mass as $p(x^* \in \mathcal{S}) = \Gamma_d([2,3]) = b^{[2,3]}_{\text{safe}}$, where $L(x^*) = \upsilon = [2,3]$. The prior estimate $B^{\text{prior}}_{\upsilon}$ of an index vector υ is either obtained by fusing all BBAs within the set $\mathcal{B}_{\upsilon}$, e.g., $B^{\text{prior}}_{[2,3]} = F(\mathcal{B}_{[2,3]})$, or set to an initial estimate, e.g., $B^{\text{prior}}_{[3,6]} = B_{\text{ini}}$.

training data with distant simplified states, e.g., y^1 and y^3 in Fig. 3. In general, the simplified state y is chosen to be 2-D or 3-D, i.e., $y \in \mathbb{R}^{n_y}$ with $n_y = 2$ or $n_y = 3$. In this article, we set $n_y = 2$.

Note that t-SNE only determines the values of simplified states but gives no expression of the state mapping $y = \Psi(x)$. Therefore, to identify the state mapping $y = \Psi(x)$, we learn a function approximator using the values of simplified states $y^1, \ldots, y^{k_t}$ obtained from t-SNE and the original system states $x^1_{\text{sim}}, \ldots, x^{k_t}_{\text{sim}}$ contained in the training dataset $\mathcal{D}_{\text{train}}$. This function approximator, e.g., we use a neural network in this article, is then utilized to represent the state mapping $y = \Psi(x) = \text{NN}(x)$.

Remark 3: Different forms of function approximator, for instance, a Gaussian process, can be used to describe the state mapping $y = \Psi(x)$. The selection of the function approximator depends mainly on the available training data.

Due to the approximation error in the function approximator, some original system states x may have slightly different values in their simplified states y when comparing the initial realization obtained from t-SNE with the one computed from the learned state mapping $y = \Psi(x)$ (for example, see the simulations in Section VI-B). Hence, to reduce the influence of this issue on deriving the low-dimensional representation of the safe region $\mathcal{S}_y$, we compute the values of simplified states $y^1, \ldots, y^{k_t}$ once again with the learned state mapping. This final realization of the low-dimensional safety feature is then used for formulating the SAF $\Gamma(y)$.

B. Belief Function Theory and DSAF

Once the state mapping $y = \Psi(x)$ is determined, we are able to generate the SAF $\Gamma(y)$ using the training dataset $\mathcal{D}_{\text{train}}$. However, due to the limited size of the training data, it is difficult to construct the SAF $\Gamma(y)$ over the continuous simplified state space $\mathcal{Y}$. Therefore, we discretize the simplified state space $\mathcal{Y}$.

The range of the simplified state space $\mathcal{Y}$ is determined by the maximum and minimum values of the simplified states $y^1, \ldots, y^{k_t}$ in each dimension. We then discretize the simplified state space $\mathcal{Y}$ into grid cells with a predefined step size. Each grid cell is assigned an index vector $\upsilon \in \mathbb{Z}^2_+$ to indicate its position in the simplified state space $\mathcal{Y}$; for example, $\upsilon = [2,3]$ refers to the grid cell that is located at the second row and third column (see Fig. 4). A locating function is defined as follows.

Definition 4: By locating the simplified state $y = \Psi(x)$ for an original system state x in the simplified state space $\mathcal{Y}$, the locating function $L(x)$ returns the index vector υ of the grid cell that it belongs to.

By assuming that the SAF $\Gamma(y)$ is constant in each grid cell, we obtain a DSAF $\Gamma_d(\upsilon)$ that we will have to define. Then, instead of using the simplified state y, the safety of an original system state x is estimated by way of the index vector υ as

$$p(x \in \mathcal{S}) = \Gamma_d(\upsilon)|_{\upsilon = L(x)} \sim [0, 1]. \tag{16}$$

In general, the DSAF $\Gamma_d(\upsilon)$ for an index vector υ can be approximated by the number of safe and unsafe original system states x that map to the corresponding grid cell, i.e., $L(x) = \upsilon$. However, due to the high-dimensional original system state space, it is, in most cases, infeasible to acquire a sufficient amount of data to derive an accurate estimate. To solve this problem, we propose using belief function theory [33] to describe the DSAF $\Gamma_d(\upsilon)$, where the uncertainty caused by insufficiency in the data amount is considered by a subjective probability [34].

Belief function theory is a general approach to modeling epistemic uncertainty that uses a belief mass to represent the probability of the occurrence of an event. The assignment of belief masses to all possible events is denoted as the basic belief assignment (BBA). The belief mass on the entire event domain, i.e., the probability that one arbitrary event happens, indicates the subjective uncertainty of the estimate [34]. According to this, we define a BBA B_{υ} separately for each index vector υ as follows.

Definition 5: The BBA B_{υ} for an index vector υ is given as

$$B_{\upsilon} = \left(b^{\upsilon}_{\text{safe}}, b^{\upsilon}_{\text{unsafe}}, \mu^{\upsilon}\right) \tag{17}$$

which represents the belief about the value of the DSAF $\Gamma_d(\upsilon)$ for the index vector υ. The belief masses $b^{\upsilon}_{\text{safe}}$ and $b^{\upsilon}_{\text{unsafe}}$ are the probabilities of the occurrence of two complementary events, i.e., $p(x \in \mathcal{S})$ and $p(x \notin \mathcal{S})$, where the original system state x has the index vector υ from the locating function $L(x)$. μ^{υ} is the subjective uncertainty that reflects the confidence level of estimating the safety. $\mu^{\upsilon} = 0$ means we believe that the estimate is absolutely correct. It holds that

$$b^{\upsilon}_{\text{safe}} + b^{\upsilon}_{\text{unsafe}} + \mu^{\upsilon} = 1 \tag{18}$$

and $b^{\upsilon}_{\text{safe}}$, $b^{\upsilon}_{\text{unsafe}}$, and μ^{υ} all lie within the interval $[0, 1]$.

Hence, the DSAF $\Gamma_d(\upsilon)$ is given by the belief masses $b^{\upsilon}_{\text{safe}}$ of the corresponding BBAs B_{υ} as

$$\Gamma_d(\upsilon) = b^{\upsilon}_{\text{safe}}. \tag{19}$$

The low-dimensional representation of the safe region $\mathcal{S}_y$ is then defined among the discretized simplified state space as

$$\mathcal{S}_y = \{v \mid \Gamma_d(v) = b^v_{\text{safe}} > p_t\} \tag{20}$$

where p_t is the predefined probability threshold. In the next subsection, we explain how to initialize the DSAF $\Gamma_d(v)$ so as to enable the application of the SRL framework on the real system.

C. Initializing the DSAF From Training Data

Since each training data provide information on the value of the DSAF $\Gamma_d(v)$, the low-dimensional representation of the safe region $\mathcal{S}_y$ is initialized using the training dataset $\mathcal{D}_{\text{train}}$. By considering each training data as a belief source, we formulate the following BBAs for all training data and later fuse them to derive an initial estimate of the DSAF $\Gamma_d(v)$.

Definition 6: The BBA B^i obtained from the ith training data D^i_{train} is defined as

$$B^i = (b^i_{\text{safe}}, b^i_{\text{unsafe}}, \mu^i). \tag{21}$$

It represents the belief about the value of the DSAF $\Gamma_d(v)$ for the index vector $v = L(x^i_{\text{sim}})$, where the belief source is the ith training data. b^i_{safe}, b^i_{unsafe} and μ^i have the same meanings as in Definition 5.

Due to the inevitable simulation-to-reality gap, we initialize the BBA of each training data with a constant uncertainty $\mu_{\text{ini}} > 0$ as

$$B^i = \begin{cases} (1-\mu_{\text{ini}}, 0, \mu_{\text{ini}}), & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) = 1 \\ (0, 1-\mu_{\text{ini}}, \mu_{\text{ini}}), & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) = 0 \end{cases} \tag{22}$$

where $i = 1, \ldots, k_t$. Since no information about the unknown term $d(x)$ is available prior to the learning process on the real system, the initial subjective uncertainties are chosen to be the same for all BBAs. Later in the online adaptation method, the subjective uncertainties are updated by using the feedback data to realize more accurate safety estimates.

For each index vector v, the BBA B_v is then estimated by using the BBAs of the training data. To achieve this, we first generate a set of BBAs $\mathcal{B}_v$ for each index vector v

$$\mathcal{B}_v = \{B^i \mid L(x^i_{\text{sim}}) = v\}. \tag{23}$$

which contains the BBAs of the training data whose original system state x_{sim} corresponds to the index vector v. The size of the set $\mathcal{B}_v$ is denoted as k_v.

Every BBA in the set $\mathcal{B}_v$ provides a belief about the value of the DSAF $\Gamma_d(v)$ for the index vector v. Hence, an estimate of the BBA B_v is derived by fusing all BBAs within the set $\mathcal{B}_v$ as

$$B^{\text{prior}}_v = \left(b^{v,\text{prior}}_{\text{safe}}, b^{v,\text{prior}}_{\text{unsafe}}, \mu^{v,\text{prior}}\right) = \begin{cases} F(\mathcal{B}_v), & \text{if } k_v \geq k_{\min} \\ B_{\text{ini}}, & \text{else} \end{cases} \tag{24}$$

where B_{ini} is an initial estimate that represents our guess about the BBA B_v when no training data are available (see Fig. 4). $F(\cdot)$ is a fusion operation among the set $\mathcal{B}_v$, which is referred to as weighted belief fusion and is defined according to [35] as

$$b^{v,\text{prior}}_{\text{safe}} = \frac{\sum_{B^i\in\mathcal{B}_v} b^i_{\text{safe}}(1-\mu^i)\prod_{\substack{B^j\in\mathcal{B}_v \\ i\neq j}}\mu^j}{\left(\sum_{B^i\in\mathcal{B}_v}\prod_{\substack{B^j\in\mathcal{B}_v \\ i\neq j}}\mu^j\right) - k_v\prod_{B^i\in\mathcal{B}_v}\mu^i} \tag{25}$$

$$b^{v,\text{prior}}_{\text{unsafe}} = \frac{\sum_{B^i\in\mathcal{B}_v} b^i_{\text{unsafe}}(1-\mu^i)\prod_{\substack{B^j\in\mathcal{B}_v \\ i\neq j}}\mu^j}{\left(\sum_{B^i\in\mathcal{B}_v}\prod_{\substack{B^j\in\mathcal{B}_v \\ i\neq j}}\mu^j\right) - k_v\prod_{B^i\in\mathcal{B}_v}\mu^i} \tag{26}$$

$$\mu^{v,\text{prior}} = \frac{\left(k_v - \sum_{B^i\in\mathcal{B}_v}\mu^i\right)\prod_{B^i\in\mathcal{B}_v}\mu^i}{\left(\sum_{B^i\in\mathcal{B}_v}\prod_{\substack{B^j\in\mathcal{B}_v \\ i\neq j}}\mu^j\right) - k_v\prod_{B^i\in\mathcal{B}_v}\mu^i}. \tag{27}$$

We refer to this estimate of the BBA B_v as the prior estimate B^{prior}_v. Since it is still likely to be imprecise if the available number of training data is too small, the fusion is performed only when the number of BBAs contained in the set $\mathcal{B}_v$ is not smaller than a minimum number $k_{\min}$. Otherwise, the prior estimate B^{prior}_v is set to the initial estimate B_{ini}. We use $B_{\text{ini}} = (0.05, 0.55, 0.4)$ in our experiments. This means that if there is very little experience available in the form of training data for one grid cell, then the respective states will initially be considered unsafe. The resulting prior estimate B^{prior}_v is a BBA that satisfies

$$b^{v,\text{prior}}_{\text{safe}} + b^{v,\text{prior}}_{\text{unsafe}} + \mu^{v,\text{prior}} = 1 \tag{28}$$

and $b^{v,\text{prior}}_{\text{safe}}$, $b^{v,\text{prior}}_{\text{unsafe}}$, and $\mu^{v,\text{prior}}$ all lie within the interval $[0, 1]$.

After computing the prior estimate B^{prior}_v for all index vectors v, we, thus, obtain a prior DSAF $\Gamma^{\text{prior}}_d(v)$

$$\Gamma^{\text{prior}}_d(v) = b^{v,\text{prior}}_{\text{safe}} \tag{29}$$

which delivers an estimate of the DSAF $\Gamma_d(v)$ that is derived from the training data. The low-dimensional representation of the safe region $\mathcal{S}_y$ is then initialized by letting $\Gamma_d(v) = \Gamma^{\text{prior}}_d(v)$. In Section V, we propose an online adaptation method to update the DSAF $\Gamma_d(v)$ using feedback data, to account for the unknown part of the system dynamics $d(x)$.

V. Online Adaptation of the Safety Assessment Function

In the early learning phase with the real system, the prior DSAF $\Gamma^{\text{prior}}_d(v)$ allows a rough estimate of the safety of an original system state. During the learning process, the feedback data are used to update the DSAF $\Gamma_d(v)$ to achieve more accurate safety estimates. Each update iteration of the DSAF $\Gamma_d(v)$ consists of three steps. First, we modify the prior DSAF $\Gamma^{\text{prior}}_d(v)$ by revising the subjective uncertainties of the BBAs of the training data. Second, we compute a feedback DSAF $\Gamma^{\text{feedback}}_d(v)$ using the feedback data. Third, the updated DSAF $\Gamma_d(v)$ is obtained by fusing the prior and feedback DSAFs. Note that each time the corrective controller $K(x)$ is activated for the real system, we obtain new feedback data. Hence, the size of the feedback dataset $\mathcal{D}_{\text{feedback}}$ increases incrementally during the learning process. For simplicity, we consider the feedback dataset $\mathcal{D}_{\text{feedback}}$ of size k_f in this section. Details of the online adaptation method are given in the following.

A. Update of the Prior DSAF With Feedback Data

The prior DSAF $\Gamma_d^{\text{prior}}(v)$ is constructed using the training dataset $\mathcal{D}_{\text{train}}$, in which the uncertainty caused by the unknown term $d(x)$ is represented by the subjective uncertainty μ^i of each BBA B^i. Hence, the update of the prior DSAF $\Gamma_d^{\text{prior}}(v)$ will now modify the subjective uncertainties by accounting for new information given by feedback data. For this, we assume that original system states that are in close proximity to each other most probably have similar safety properties.

Assumption 2: The probability $p(s_{\text{real}}(x^1) = s_{\text{real}}(x^2))$ that two original system states x^1 and x^2 have the same safety property on the real system is inversely proportional to their Euclidean distance in the original state space $||x^1 - x^2||$.

In addition, we define a function $P(x)$ to quantify the similarity with respect to the safety of nominal and real system trajectories that start in the same initial original system state x

$$P(x) = p(s_{\text{sim}}(x) = s_{\text{real}}(x)) \sim [0, 1]. \tag{30}$$

It represents the probability that for a given original system state x, its safety label $s_{\text{sim}}(x)$ obtained with the nominal system is the same as the safety label $s_{\text{real}}(x)$ obtained with the real system. Then, according to Assumption 2, if we observe an original system state x that has the same safety property both in simulation and in reality, it is likely that other original system states that are close to the observed state will also show the same safety property.

In order to predict the value of the function $P(x)$, we approximate it with a Gaussian process regression (GPR) model $P(x) = \text{GP}(x)$. For each original system state x_{real} contained in the feedback dataset $\mathcal{D}_{\text{feedback}}$, we examine its safety label $s_{\text{sim}}(x_{\text{real}})$ in simulation. This leads to a set of samples $\{P(x_{\text{real}}^1), \dots, P(x_{\text{real}}^{k_f})\}$ for the function $P(x)$, in which

$$P\left(x_{\text{real}}^i\right) = \begin{cases} 1, & \text{if } s_{\text{sim}}\left(x_{\text{real}}^i\right) = s_{\text{real}}\left(x_{\text{real}}^i\right) \\ 0, & \text{if } s_{\text{sim}}\left(x_{\text{real}}^i\right) \neq s_{\text{real}}\left(x_{\text{real}}^i\right) \end{cases} \tag{31}$$

for $i = 1, \dots, k_f$. Hence, the GPR model $\text{GP}(x)$ is trained with the sets $\{x_{\text{real}}^1, \dots, x_{\text{real}}^{k_f}\}$ and $\{P(x_{\text{real}}^1), \dots, P(x_{\text{real}}^{k_f})\}$, which are obtained from the current feedback dataset $\mathcal{D}_{\text{feedback}}$.

Remark 4: If the real system is a real-world dynamical system, then it is usually difficult to test the corrective controller $K(x)$ with arbitrary initial original system states x in reality, since there is a high risk of encountering unsafe behaviors. However, in contrast, the simulation can be initialized with any original system state x_{real} contained in the feedback data, which then makes it possible to approximate the function $P(x)$.

The trained GPR model $\text{GP}(x)$ is then used to update the BBA B^i of each training data. The general motivation is that we decrease the subjective uncertainty μ^i if we are confident about the reliability of this training data. Hence, for the ith training data, we compute a predicted mean value of the function $P(x_{\text{sim}}^i)$, denoted as p_{mean}^i, from the GPR model $\text{GP}(x)$, along with a corresponding standard deviation p_{std}^i of the predicted value. Since a low value of the standard deviation p_{std}^i means we have observed enough feedback data to make a reliable prediction, we only update the BBA B^i if the standard

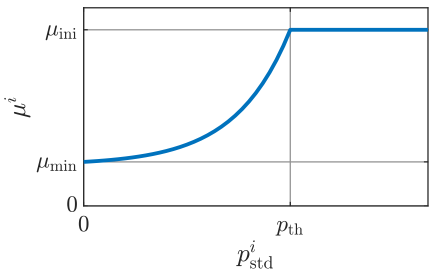


Fig. 5. As given in (33), the subjective uncertainty μ^i in the BBA B^i of the ith training data is determined using the corresponding standard deviation p_{std}^i obtained from the GPR model $\text{GP}(x)$.

deviation p_{std}^i is smaller than a predefined threshold p_{th}

$$B^i = \begin{cases} \left(p_{\text{mean}}^i\left(1-\mu^i\right), \left(1-p_{\text{mean}}^i\right)\left(1-\mu^i\right), \mu^i\right), \\ \qquad \text{if } p_{\text{std}}^i \le p_{\text{th}} \text{ and } s_{\text{sim}}\left(x_{\text{sim}}^i\right) = 1 \\ \left(\left(1-p_{\text{mean}}^i\right)\left(1-\mu^i\right), p_{\text{mean}}^i\left(1-\mu^i\right), \mu^i\right), \\ \qquad \text{if } p_{\text{std}}^i \le p_{\text{th}} \text{ and } s_{\text{sim}}\left(x_{\text{sim}}^i\right) = 0 \end{cases} \tag{32}$$

with the new subjective uncertainty μ^i calculated as

$$\mu^i = \frac{\mu_{\text{ini}} - \mu_{\min}}{\alpha^{p_{\text{th}}} - 1}\left(\alpha^{p_{\text{std}}^i} - 1\right) + \mu_{\min} \tag{33}$$

where μ_{ini} is the same initial subjective uncertainty as that given in (22) [see Fig. 5 for a graphical representation of (33)]. BBAs B^i with $p_{\text{std}}^i > p_{\text{th}}$ remain unchanged, as in (22). Such an update of the BBA B^i considers the predicted value of the function $P(x_{\text{sim}}^i)$ and the reliability of this prediction at the same time.

Equation (33) is designed by considering two aspects: first, the subjective uncertainty μ^i is set equal to μ_{ini} when $p_{\text{std}}^i \ge p_{\text{th}}$. This means that, in this case, we do not have the confidence to update the BBA B^i, as not enough information is observed from the feedback data; and second, due to the inevitable reality gap, the subjective uncertainty μ^i maintains a minimum uncertainty $\mu_{\min}$ even when the standard deviation p_{std}^i is 0. We use the exponential form such that the decrease in μ^i is faster when the standard deviation p_{std}^i is near the threshold p_{th}. The parameter $\alpha > 1$ determines the decay rate and is selected by considering the actual learning task.

Note that for the same training data, the relationship between the standard deviation p_{std}^i and the threshold p_{th} can change during the learning process. For example, we might obtain $p_{\text{std}}^i \le p_{\text{th}}$ in the current update iteration, but in the next update iteration it changes to $p_{\text{std}}^i > p_{\text{th}}$. This happens primarily when we first observe a safe original system state but followed by a nearby unsafe state, such that the safety of the states in between these two observed states becomes uncertain. In such cases, we set the BBA B^i back to the initial BBA given in (22).

Once the BBAs B^i of all training data have been updated with the up-to-date feedback dataset $\mathcal{D}_{\text{feedback}}$, the prior estimate B_v^{prior} for each index vector v is recomputed using (24). This results in an updated prior DSAF $\Gamma_d^{\text{prior}}(v)$, which is used later for revising the DSAF $\Gamma_d(v)$.

B. Feedback DSAF

The feedback data contain the information about the real safety properties of different original system states x. To fully utilize this valuable information, we construct an additional DSAF, denoted as the feedback DSAF $\Gamma_d^{\text{feedback}}(\upsilon)$, using the feedback dataset $\mathcal{D}_{\text{feedback}}$.

As the amount of data is insufficient, we also consider the estimate obtained from the feedback data as a subjective probability [26]. Then, as with the prior estimate $B_\upsilon^{\text{prior}}$, we formulate another estimate of the BBA B_υ for each index vector υ as

$$B_\upsilon^{\text{feedback}} = \left(b_{\text{safe}}^{\upsilon,\text{feedback}}, b_{\text{unsafe}}^{\upsilon,\text{feedback}}, \mu^{\upsilon,\text{feedback}}\right) \tag{34}$$

which is referred to as the feedback estimate $B_\upsilon^{\text{feedback}}$.

For each index vector υ, the feedback estimate $B_\upsilon^{\text{feedback}}$ is determined by the number of safe and unsafe feedback data that correspond to this grid cell. By sorting the feedback dataset $\mathcal{D}_{\text{feedback}}$ with the locating function $L(x)$, we denote the number of safe feedback data that have the index vector υ from the locating function, i.e., $L(x_{\text{real}}) = \upsilon$ and $s_{\text{real}}(x_{\text{real}}) = 1$, as k_{safe}^υ (and $k_{\text{unsafe}}^\upsilon$ for the number of unsafe feedback data). If at least one feedback data is available for the index vector υ, i.e., $k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon \geq 1$, we compute the feedback estimate $B_\upsilon^{\text{feedback}}$ as follows:

$$b_{\text{safe}}^{\upsilon,\text{feedback}} = \frac{k_{\text{safe}}^\upsilon}{k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon}\left(1 - \mu^{\upsilon,\text{feedback}}\right) \tag{35}$$

$$b_{\text{unsafe}}^{\upsilon,\text{feedback}} = \frac{k_{\text{unsafe}}^\upsilon}{k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon}\left(1 - \mu^{\upsilon,\text{feedback}}\right) \tag{36}$$

$$\mu^{\upsilon,\text{feedback}} = \beta\exp\left(-\gamma\left(k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon - 1\right)\right). \tag{37}$$

The subjective uncertainty $\mu^{\upsilon,\text{feedback}}$ decreases if more feedback data are observed for the index vector υ. It satisfies that, if a sufficient number of feedback data are obtained, the subjective uncertainty $\mu^{\upsilon,\text{feedback}}$ approaches 0. In such a case, the belief masses $b_{\text{safe}}^{\upsilon,\text{feedback}}$ and $b_{\text{unsafe}}^{\upsilon,\text{feedback}}$ can be considered as the actual probabilities. The parameters β and γ define the initial value and the decay rate of the subjective uncertainty $\mu^{\upsilon,\text{feedback}}$, respectively. If no feedback data are observed for the index vector υ, we set the feedback estimate $B_\upsilon^{\text{feedback}}$ to an empty BBA $B_\varnothing$ defined as $B_\upsilon^{\text{feedback}} = B_\varnothing = (0, 0, 1)$, which indicates that no safety estimate can be made.

Using the feedback estimate $B_\upsilon^{\text{feedback}}$, we obtain the following feedback DSAF $\Gamma_d^{\text{feedback}}(\upsilon)$:

$$\Gamma_d^{\text{feedback}}(\upsilon) = b_{\text{safe}}^{\upsilon,\text{feedback}} \tag{38}$$

which represents the estimate of the DSAF $\Gamma_d(\upsilon)$ derived from the feedback data only. In Section V-C, we fuse the feedback DSAF $\Gamma_d^{\text{feedback}}(\upsilon)$ with the updated prior DSAF $\Gamma_d^{\text{prior}}(\upsilon)$ to derive a more accurate DSAF $\Gamma_d(\upsilon)$.

C. Fusion of Prior and Feedback DSAFs

The prior and feedback DSAFs both provide beliefs about safety by using different datasets as their belief source. To update the DSAF $\Gamma_d(\upsilon)$, we fuse these two functions using weighted belief fusion as given in (25)–(27). This leads to a fused estimate B_υ^{fuse} for each index vector υ

$$B_\upsilon^{\text{fuse}} = \left(b_{\text{safe}}^{\upsilon,\text{fuse}}, b_{\text{unsafe}}^{\upsilon,\text{fuse}}, \mu^{\upsilon,\text{fuse}}\right) \tag{39}$$

which is computed as

$$B_\upsilon^{\text{fuse}} = \begin{cases} F\left(\left\{B_\upsilon^{\text{prior}}, B_\upsilon^{\text{feedback}}\right\}\right), & \text{if } B_\upsilon^{\text{feedback}} \neq B_\varnothing \\ B_\upsilon^{\text{prior}}, & \text{if } B_\upsilon^{\text{feedback}} = B_\varnothing. \end{cases} \tag{40}$$

If the feedback estimate $B_\upsilon^{\text{feedback}}$ is nonempty, we find the fused estimate B_υ^{fuse} through weighted belief fusion $F(\cdot)$ of the set $\{B_\upsilon^{\text{prior}}, B_\upsilon^{\text{feedback}}\}$. Otherwise, we set the fused estimate B_υ^{fuse} equal to the prior estimate $B_\upsilon^{\text{prior}}$.

The fused estimate B_υ^{fuse} fulfills the following property, which is also given in [26].

Proposition 1: If the number of feedback data approaches infinity, the fused estimate B_υ^{fuse} becomes the actual probabilities, and the prior estimate $B_\upsilon^{\text{prior}}$ has no effect in making safety estimates.

Proof: Proposition 1 is justified by the following:

$$\lim_{k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon \to \infty} b_{\text{safe}}^{\upsilon,\text{fuse}} = b_{\text{safe}}^{\upsilon,\text{feedback}} \tag{41}$$

$$\lim_{k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon \to \infty} b_{\text{unsafe}}^{\upsilon,\text{fuse}} = b_{\text{unsafe}}^{\upsilon,\text{feedback}} \tag{42}$$

$$\lim_{k_{\text{safe}}^\upsilon + k_{\text{unsafe}}^\upsilon \to \infty} \mu^{\upsilon,\text{fuse}} = \mu^{\upsilon,\text{feedback}} = 0 \tag{43}$$

which are obtained by simplifying (25)–(27) with the set $\{B_\upsilon^{\text{prior}}, B_\upsilon^{\text{feedback}}\}$. □

Considering computational efficiency, the update of the DSAF $\Gamma_d(\upsilon)$ is generally performed once when every k_u feedback data are obtained, where the value of k_u is selected according to the actual learning task. In each update iteration (indexed by number N, see Section VI-C), we first use the up-to-date feedback dataset $\mathcal{D}_{\text{feedback}}$ to update the prior DSAF $\Gamma_d^{\text{prior}}(\upsilon)$ and to construct the feedback DSAF $\Gamma_d^{\text{feedback}}(\upsilon)$. Then, the fused estimate B_υ^{fuse} is computed from these two functions for each index vector υ. The updated DSAF $\Gamma_d(\upsilon)$ is, thus, obtained using the fused estimate B_υ^{fuse} as

$$\Gamma_d(\upsilon) = b_{\text{safe}}^{\upsilon,\text{fuse}} \tag{44}$$

which also gives the latest low-dimensional representation of the safe region $\mathcal{S}_y$ according to (20). With further feedback data, the DSAF $\Gamma_d(\upsilon)$ becomes more accurate, and more reliable safety estimates are obtained.

VI. Quadcopter Experiments

In this section, we demonstrate the proposed approach for identifying the low-dimensional representation of the safe region $\mathcal{S}_y$, using the example of a quadcopter.

A. Experimental Setup

We simulate the quadcopter using the system dynamics given in [36] with MATLAB Simulink[2] (Version R2019b) running on a laptop powered by an Intel i7-7700HQ CPU. The 12-D system state is defined as $x = [p_g, \theta_g, v_b, \omega_b]^T$,

[2]https://www.mathworks.com/products/simulink.html

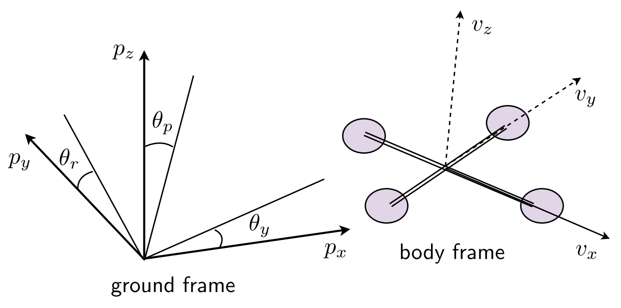


Fig. 6. System state x of a quadcopter is defined using the ground frame and the body frame.

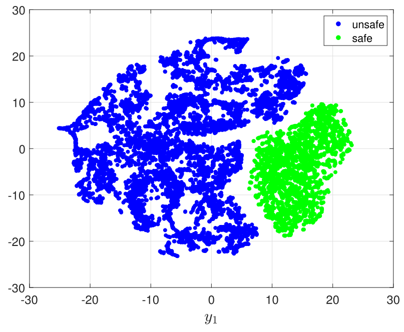
where $p_g = [p_x, p_y, p_z]^T$ and $\theta_g = [\theta_r, \theta_p, \theta_y]^T$ are the linear and angular positions defined in the ground frame, respectively, $v_b = [v_x, v_y, v_z]^T$ and $\omega_b = [\omega_r, \omega_p, \omega_y]^T$ are the linear and angular velocities defined in the body frame (see Fig. 6), respectively. The control input u consists of the four motor speeds of the quadcopter. For the nominal system model, we set the mass of the quadcopter to $m = 1$ kg and the maximal lifting force to $f = 200$ N. The safety of a given state x is determined by simulating the controlled dynamics with the corrective control $K(x)$ that starts in the initial state x and checking if the controller is able to successfully drive the quadcopter back to a hovering state without crashing. In this example, we use the PID controller given in [36] as the corrective controller $K(x)$. It stabilizes the quadcopter's height and its roll, pitch, and yaw rotations. The coefficients of the PID controller are: $K_{P,h} = 1.5$, $K_{I,h} = 0$, and $K_{D,h} = 2.5$ for the height control, and $K_{P,r} = K_{P,p} = K_{P,y} = 6$, $K_{I,r} = K_{I,p} = K_{I,y} = 0$, and $K_{D,r} = K_{D,p} = K_{D,y} = 1.75$ for the roll, pitch, and yaw rotations control, respectively.

To generate the training dataset $\mathcal{D}_{\text{train}}$, we first create $k_t = 10\,000$ original system states x. We set $p_x = p_y = 0$ and $p_z = 2$ m to leave enough space and time for the corrective controller $K(x)$. All other variables are sampled with a uniform distribution within the following range: $0 \leq \theta_r, \theta_p, \theta_y \leq 2\pi$ rad, -3 m/s $\leq v_x, v_y, v_z \leq 3$ m/s, and -10 rad/s $\leq \omega_r, \omega_p, \omega_y \leq 10$ rad/s. The training dataset $\mathcal{D}_{\text{train}}$ is then obtained by examining the performance of the corrective controller $K(x)$ for all these initial values.

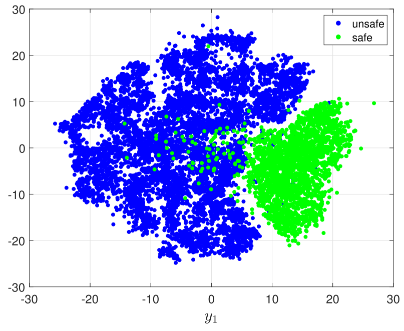
B. Identifying the Low-Dimensional Representation of the Safe Region

The initial realization of the low-dimensional safety feature, i.e., the values of simplified states $y^1, \ldots, y^{k_t}$, obtained from t-SNE is given in Fig. 7(a). We use $\delta = 0.01$ in (14) and set the perplexity and tolerance of t-SNE (see [32]) to 40 and $1e^{-4}$, respectively. The result shows that the safe and unsafe original system states are clearly separated in the 2-D simplified state space $\mathcal{Y} \subseteq \mathbb{R}^2$.

The state mapping $y = \Psi(x)$ is represented by a two-layer neural network with 128 neurons in each layer, which is trained using the initial realization of simplified states $y^1, \ldots, y^{k_t}$ and the set of original system states $\{x^1_{\text{sim}}, \ldots, x^{k_t}_{\text{sim}}\}$. By recomputing the outputs of the learned neural network, we obtain the final realization of



(a)



(b)

Fig. 7. (a) Initial realization of simplified states $y^1, \ldots, y^{k_t}$ obtained from t-SNE. The safe and unsafe training data are denoted by green and blue points, respectively. (b) Final realization of simplified states $y^1, \ldots, y^{k_t}$ obtained by recomputing with the learned neural network that represents the state mapping $y = \Psi(x) = \text{NN}(x)$.

the low-dimensional safety feature, i.e., the values of the simplified states $y^1, \ldots, y^{k_t}$, given in Fig. 7(b). Due to approximation error, certain simplified states have a slightly changed position compared with the values obtained from t-SNE. However, this does not affect the computation of the low-dimensional representation of the safe region $\mathcal{S}_y$, as the results are updated later in the online adaptation using the feedback data.

We set the simplified state space as $\{\mathcal{Y} \mid -30 \leq y_1, y_2 \leq 30\}$. By discretizing the simplified state space $\mathcal{Y}$ into grid cells with step size 1 in both y_1 and y_2, we obtain the index vector $v \in \{1, 2, \ldots, 60\}^2$. The prior DSAF $\Gamma^{\text{prior}}_d(v)$ is, thus, computed from the training dataset $\mathcal{D}_{\text{train}}$ using the index vector v. The results are given in Fig. 8(a), where the initial subjective uncertainty, the initial estimate and the minimum number are selected as $\mu_{\text{ini}} = 0.4$, $B_{\text{ini}} = (0.05, 0.55, 0.4)$, and $k_{\min} = 3$, respectively. Depending on the number of safe and unsafe training data in each grid cell, the prior DSAF $\Gamma^{\text{prior}}_d(v)$ estimates the probability $p(x \in \mathcal{S})$ for original system states x that take the index vector v from the locating function $L(x)$. In Fig. 8(i), the DSAF $\Gamma_d(v)$ is initialized by the prior DSAF $\Gamma^{\text{prior}}_d(v)$. In Section VI-C, we demonstrate the update process of the DSAF $\Gamma_d(v)$ using the proposed online adaptation method.

C. Updating the Low-Dimensional Representation

To simulate a mismatch between the nominal and the real systems, we set the mass and the maximal lifting force of

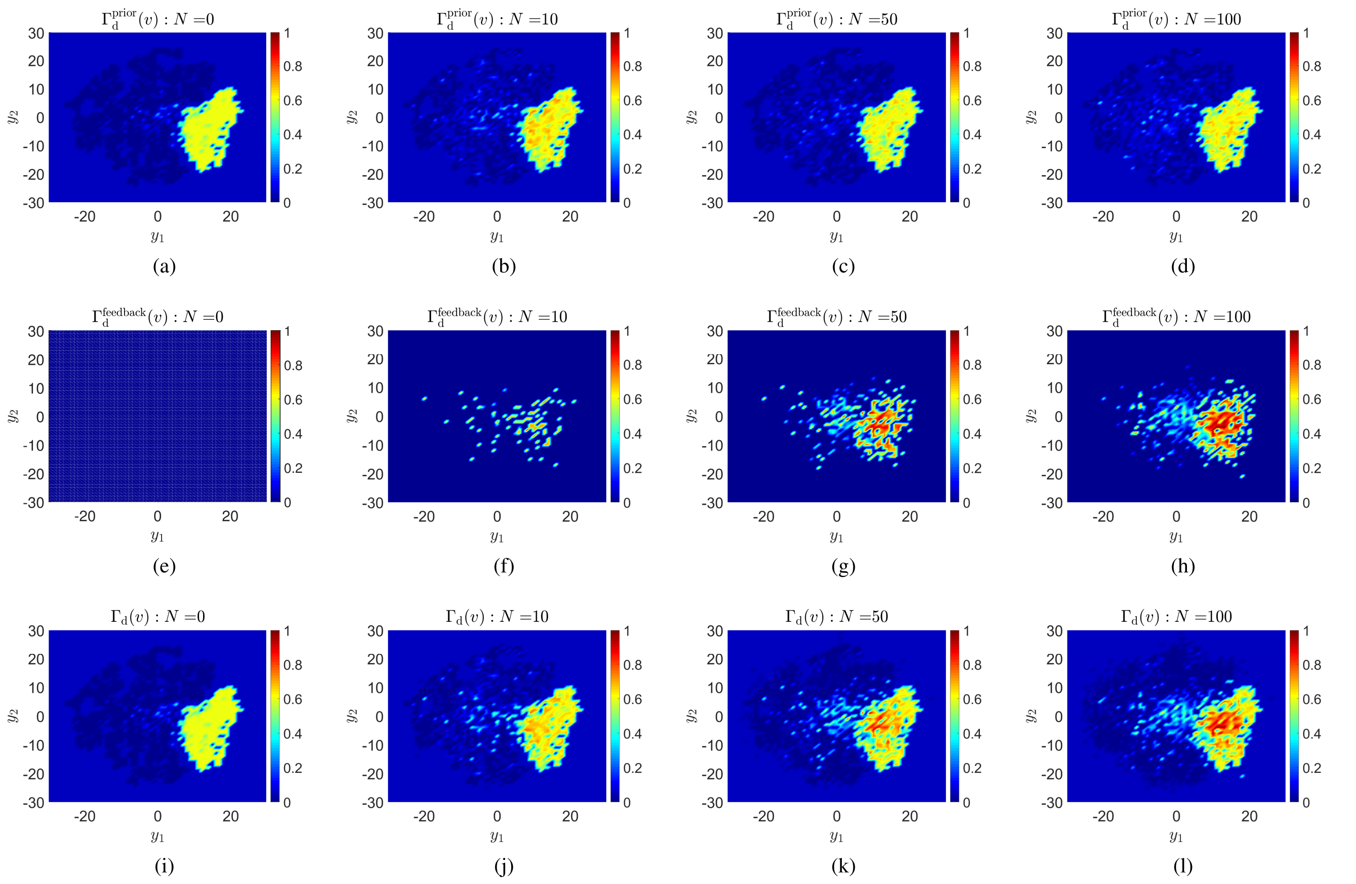


Fig. 8. Results of the online adaptation. (a)–(d) Prior DSAF $\Gamma_d^{\text{prior}}(v)$ in different update iterations N. $N = 0$ refers to the initialization prior to the online adaptation. The values of the safety estimates are represented by different colors. (e)–(h) Feedback DSAF $\Gamma_d^{\text{feedback}}(v)$ in different update iterations N. (i)–(l) DSAF $\Gamma_d(v)$ in different update iterations N.

the real system to $m = 0.8$ kg and $f = 145$ N, respectively. To eliminate the influence of a specific learning task or algorithm and focus on illustrating the update process, the feedback dataset $\mathcal{D}_{\text{feedback}}$ is obtained by randomly selecting states x_{real} where the corrective controller $K(x)$ is activated, such that the entire original system state space can be visited.

The following parameters are used in the online adaptation method: $\mu_{\min} = 0.1$, $p_{\text{th}} = 0.3$, $\alpha = 3e^5$, $\beta = 0.3$, and $\gamma = 0.4$. The GPR model $\text{GP}(x)$ uses a squared exponential kernel. To demonstrate the online update process, we collect the feedback data one by one and incrementally extend the feedback dataset $\mathcal{D}_{\text{feedback}}$. The DSAF $\Gamma_d(v)$ is updated once when every $k_u = 20$ feedback data are obtained.

The results of the online adaptation are given in Fig. 8. Prior to the update (update iteration $N = 0$), the DSAF $\Gamma_d(v)$ is initialized as the prior DSAF $\Gamma_d^{\text{prior}}(v)$, while the feedback DSAF $\Gamma_d^{\text{feedback}}(v)$ is constructed using the empty BBA $B_\varnothing$ [see Fig. 8(a), (e), and (i)]. Once the learning procedure has started, we collect the feedback data incrementally. In the early updating phase, e.g., update iteration $N = 10$, the DSAF $\Gamma_d(v)$ is mainly determined by the prior DSAF $\Gamma_d^{\text{prior}}(v)$. The subjective uncertainties of each training data are modified using the feedback data, where we become confident about the safety of certain training data when we observe a nearby feedback data that has the same safety property. Since the amount of feedback data is insufficient for providing a reliable safety estimate, the feedback DSAF $\Gamma_d^{\text{feedback}}(v)$ has a smaller effect on the computation of the low-dimensional representation of the safe region $\mathcal{S}_y$ [see Fig. 8(b), (f), and (j)].

When more feedback data are available, e.g., update iteration $N = 50$, the feedback DSAF $\Gamma_d^{\text{feedback}}(v)$ is able to provide more accurate safety estimates; hence, its influence on the DSAF $\Gamma_d(v)$ also becomes more significant. Due to the high dimensionality of the original system state x and the limited amount of feedback data, it is difficult to acquire an estimate with high confidence from the GPR model $\text{GP}(x)$. As a result, changes are marginal in the prior DSAF $\Gamma_d^{\text{prior}}(v)$ [see Fig. 8(c), (g), and (k)]. With even more feedback data, e.g., update iteration $N = 100$, the DSAF $\Gamma_d(v)$ is able to provide reliable estimates about the probability $p(x \in \mathcal{S})$ for each index vector v. While the prior and feedback DSAFs are updated accordingly, the DSAF $\Gamma_d(v)$ represents the actual low-dimensional representation of the safe region $\mathcal{S}_y$ under the unknown part of the system dynamics $d(x)$ [see Fig. 8(d), (h), and (l)].

D. Comparison With Physically Inspired Model Order Reduction

We compare the proposed approach with the physically inspired model order reduction presented in [26] in terms of the representation power of the identified low-dimensional

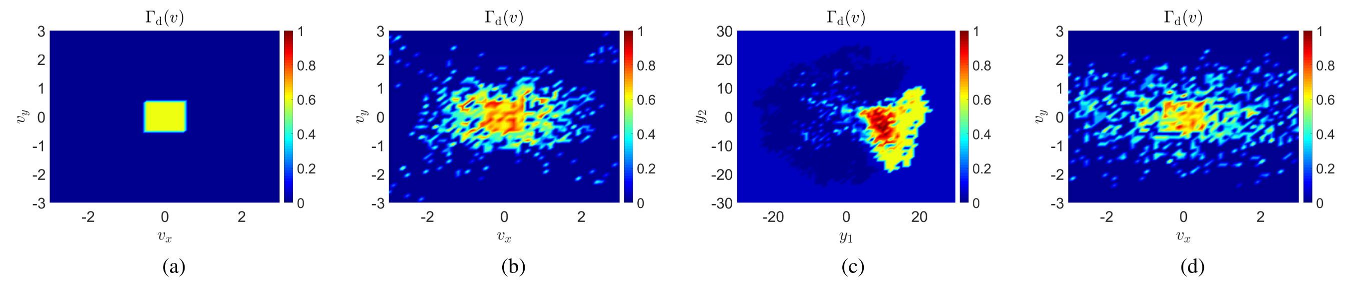


Fig. 9. Comparison with physically inspired model order reduction. (a) For physically inspired model order reduction, the DSAF $\Gamma_d(v)$ is initialized conservatively. (b) and (c) DSAFs $\Gamma_d(v)$ obtained by using physically inspired model order reduction and the proposed approach, respectively. The feedback dataset $\mathcal{D}'_{\text{feedback}}$ is used for the update. (d) DSAF $\Gamma_d(v)$ obtained by using physically inspired model order reduction and the feedback dataset $\mathcal{D}_{\text{feedback}}$.

representation of the safe region $\mathcal{S}_y$, i.e., how well the safe and unsafe states are separated. To do this, we compute another DSAF $\Gamma_d(v)$ using physical features. As in [26], the low-dimensional safety feature, i.e., the simplified state y, is selected for the velocities in the x- and y-directions $y = [v_x, v_y]^T$. To avoid any dangerous behavior in early learning phase, the low-dimensional representation of the safe region $\mathcal{S}_y$ is initialized conservatively [26] by setting $\Gamma_d(v) = 0.6$ for grid cells that satisfy $-0.5 \le v_x, v_y \le 0.5$ [see Fig. 9(a)].

As the learning task in [26] is relatively simple, the exploration in the original system state space is limited to a small subspace around the origin (see Section VII-A for more discussions on this point). Therefore, to make a fair comparison, we also generate another feedback dataset $\mathcal{D}'_{\text{feedback}}$ that has the same size as the dataset $\mathcal{D}_{\text{feedback}}$. However, instead of the complete original system state space given in Section VI-A, the states x_{real} in the set $\mathcal{D}'_{\text{feedback}}$ are sampled from a smaller state space, where the ranges of angular positions and angular velocities are changed to $-(1/3)\pi \le \theta_r, \theta_p, \theta_y \le (1/3)\pi$ rad and -3 rad/s $\le \omega_r, \omega_p, \omega_y \le 3$ rad/s, respectively.

We first compare the performance of both approaches by considering a small state space, i.e., the feedback dataset $\mathcal{D}'_{\text{feedback}}$ is used for the update. The results show that, in this case, physical features are able to provide reasonable predictions about safety, i.e., the safe and unsafe regions are separated [see Fig. 9(b)]. Meanwhile, the proposed approach also produces a satisfying result with marginally better separation between the safe and unsafe states [see Fig. 9(c)].

However, if the learning task becomes more complex, the complete state space usually has to be explored to enable an optimal policy to be found. To simulate this scenario, we also update the initial DSAF $\Gamma_d(v)$ using the feedback dataset $\mathcal{D}_{\text{feedback}}$. As seen in Fig. 9(d), when considering the entire original system state space, it is difficult to make reliable safety estimates based only on physical features. The boundary between safe and unsafe regions becomes unclear, and there are numerous grid cells that lead to a safety estimate close to 0.5. In contrast, the proposed approach is still able to find a representative low-dimensional representation of the safe region $\mathcal{S}_y$ for the complete state space. As the identified simplified state y can describe the safety of the original system states x more precisely, a satisfying separation between the safe and unsafe regions is achieved [see Fig. 8(l)] and more useful safety estimates are obtained. The independence of the size of the state space indicates the possibility of implementing the proposed approach on different learning tasks, which, in turn, increases the applicability of the SRL framework.

VII. Discussion

In this work, we propose a general approach for efficiently identifying a low-dimensional representation of the safe region. Two important aspects of the proposed approach are discussed in this section.

A. Relevance to Different SRL Tasks

In [26], the SRL framework utilizes the low-dimensional representation of the safe region $\mathcal{S}_y$ that is obtained using physically inspired model order reduction. Such a low-dimensional representation is useful when the learning task is relatively simple, e.g., teaching a quadcopter to fly forward, as given in [26], such that a satisfying control policy can be found without requiring an extensive exploration in the original state space. Since, in this case, the system state is likely to stay in a substate space near the origin, physical features are able to provide reliable safety estimates. However, when the learning task becomes more difficult, e.g., the quadcopter needs to track a complex 3-D trajectory, the learning algorithm in general has to explore a large portion of the state space to find an optimal policy. Under these circumstances, at least a rough safety assessment of the complete state space is needed. Unfortunately, being restricted by the representation power, the physically inspired low-dimensional representation of the safe region $\mathcal{S}_y$ fails to provide useful safety estimates when considering the entire state space. Hence, the performance of the SRL framework is affected.

Therefore, to overcome this problem, this article proposes a data-driven approach for identifying a low-dimensional representation of the safe region $\mathcal{S}_y$ that is able to make more precise predictions about safety. Meaningful safety estimates are even obtained for the entire original state space. This not only gives the learning algorithm more flexibility in choosing its actions to find the optimal policy but also indicates the applicability of the proposed approach to more complex learning tasks.

B. Strengths and Limitations

The presented approach has three particular strengths. First, it finds a low-dimensional representation of the safe region

$\mathcal{S}_y$ that allows safe and unsafe states to be clearly separated for large portions of a high-dimensional state space; see also Section VI-D. Second, the effort required for identifying the low-dimensional representation of the safe region $\mathcal{S}_y$ is low. While, for instance, physically inspired model order reduction usually needs a comprehensive analysis of the system dynamics, the proposed approach relies solely on training data that can be collected efficiently even for complex dynamical systems through parallel computing and a suitable simulation environment. Third, it fully utilizes the information contained in the feedback data using two DSAFs. Hence, the update can be performed with little feedback data while providing a satisfying result.

However, the performance of the identified low-dimensional representation of the safe region $\mathcal{S}_y$ is affected by the quality of the nominal system, i.e., the magnitude of the discrepancy between the nominal and the real systems. While the state mapping $y = \Psi(x)$ is determined using only training data, the online adaptation method attempts to find an accurate DSAF $\Gamma_d(v)$ based on the learned low-dimensional safety feature. If the reality gap is too large, then it is possible that the learned safety feature is not sufficiently representative and we might therefore observe more grid cells with final safety estimates that are close to 0.5, i.e., $\Gamma_d(v) \approx 0.5$, which are less useful for guiding the learning process. In general, if the nominal system is assumed to be unreliable, a high probability threshold p_t should be used for constructing the low-dimensional representation of the safe region $\mathcal{S}_y$ [see (20)], such that the learning process becomes more conservative for keeping the system safe. However, we usually consider the unknown system dynamics $d(x)$ as bounded within a reasonable range, since it makes less sense to use a dissimilar nominal system to predict the behavior of the real system. To further generalize the proposed approach, more studies are required to quantify the influence of the simulation-to-reality gap on the reliability of the obtained safety estimates.

VIII. Conclusion

To apply SRL to complex dynamical systems, this article proposes a novel data-driven approach to identify a low-dimensional representation of the safe region for realizing a general SRL framework. Using a nominal system model that predicts the behavior of the real system, we first collect training data about the safety of different system states. Then, by computing the probabilistic similarities between each training data using a data-driven method, an initial low-dimensional representation of the safe region is obtained. To compensate for the mismatch between the nominal and the real systems, an efficient online adaptation method based on belief function theory is also proposed to update the low-dimensional representation of the safe region by accounting for the real-system behavior. Experimental results show that, compared with the previous work, a more reliable and representative low-dimensional representation of the safe region is found using the proposed approach. However, our approach has the limitation that its performance is affected by the magnitude of discrepancy between the nominal and real systems. If the reality gap is assumed to be large, then it is likely that a less meaningful low-dimensional representation of the safe region will be obtained.

For future work, we intend to combine the data-driven method with model-based model order reduction techniques to find an approach that is more robust to the simulation-to-reality gap when identifying the low-dimensional representation of the safe region. Moreover, we also plan to investigate the possibility of quantifying the similarity between different dynamical systems, such that the learned safety feature can be generalized from one system to other similar systems. How the similarity between dynamical systems will be measured is, however, still an open research problem.

References

[1] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Jul. 2017.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, Apr. 2016.

[3] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, May 2016, pp. 1329–1338.

[4] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 42, pp. 1437–1480, Aug. 2015.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2006, pp. 1–8.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[7] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," 2016, *arXiv:1610.03518*. [Online]. Available: http://arxiv.org/abs/1610.03518

[8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," 2017, *arXiv:1702.02284*. [Online]. Available: http://arxiv.org/abs/1702.02284

[9] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Aug. 2017, pp. 22–31.

[10] Y. Shen, M. J. Tobia, T. Sommer, and K. Obermayer, "Risk-sensitive reinforcement learning," *Neural Comput.*, vol. 26, no. 7, pp. 1298–1328, Jul. 2014.

[11] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, no. 4, pp. 803–832, Dec. 2002.

[12] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2018, pp. 8103–8112.

[13] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," *Int. J. Robot. Res.*, vol. 35, no. 13, pp. 1547–1563, May 2016.

[14] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. Robot., Sci. Syst. (RSS)*, Jun. 2016, pp. 171–181.

[15] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[16] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Safe reinforcement learning using robust action governor," in *Proc. 3rd Conf. Learn. Dyn. Control (L4DC)*, Jun. 2021, pp. 1093–1104.

[17] T. M. Moldovan and P. Abbeel, "Safe exploration in Markov decision processes," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, Jun. 2012, pp. 1451–1458.

[18] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019.

[19] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[20] S. Bansal, M. Chen, S. Herbert, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE Conf. Decision Control*, Dec. 2017, pp. 2242–2253.

[21] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 4661–4666.

[22] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2017, pp. 908–919.

[23] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8550–8556.

[24] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. 2nd Conf. Robot Learn. (CoRL)*, Oct. 2018, pp. 561–591.

[25] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3019–3026, Apr. 2020.

[26] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "A general framework to increase safety of learning algorithms for dynamical systems based on region of attraction estimation," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1472–1490, Oct. 2020.

[27] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*. New York, NY, USA: Springer, 2008.

[28] A. Marco *et al.*, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1557–1563.

[29] M. Z. Romdlony and B. Jayawardhana, "Stabilization with guaranteed safety using control Lyapunov–Barrier function," *Automatica*, vol. 66, pp. 39–47, Apr. 2016.

[30] M. Sobotka, J. Wolff, and M. Buss, "Invariance controlled balance of legged robots," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2007, pp. 3179–3186.

[31] A. A. Ahmadi and A. Majumdar, "DSOS and SDSOS optimization: More tractable alternatives to sum of squares and semidefinite optimization," *SIAM J. Appl. Algebra Geometry*, vol. 3, no. 2, pp. 193–230, Apr. 2019.

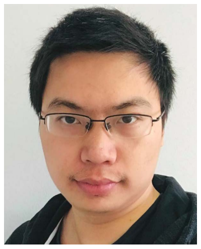
[32] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[33] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ, USA: Princeton Univ. Press, 1976.

[34] A. Jøsang, *Subjective Logic*. New York, NY, USA: Springer, 2016.

[35] A. Jøsang, "Categories of belief fusion," *J. Adv. Inf. Fusion*, vol. 13, no. 2, pp. 235–254, Dec. 2018.

[36] T. Luukkonen, "Modelling and control of quadcopter," School Sci., Aalto Univ., Espoo, Finland, Tech. Rep. Mat-2.4108, Aug. 2011.



Zhehua Zhou (Graduate Student Member, IEEE) received the B.E. degree in mechatronics engineering from Tongji University, Shanghai, China, in 2014, and the M.Sc. degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany, in 2017, where he is currently pursuing the Ph.D. degree in learning-based control and robotics with the Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering.

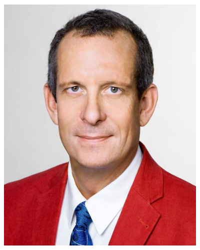
His research interests include optimal control, learning-based control, and applications to robotics.

Ozgur S. Oguz (Member, IEEE) received the B.Sc. and M.Sc. *(summa cum laude)* degrees in computer science from Koc University, Istanbul, Turkey, in 2007 and 2010, respectively, and the Ph.D. degree *(summa cum laude)* from the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany, in 2018.

He is currently a Post-Doctoral Researcher with the Machine Learning and Robotics Laboratory, University of Stuttgart, Stuttgart, Germany, and the Max Planck Institute for Intelligent Systems, Stuttgart. His research interests are developing autonomous systems that are able to reason about their states of knowledge, take sequential decisions to realize a goal, and simultaneously learn to improve their causal physical reasoning and manipulation skills.

Marion Leibold (nee Sobotka) (Member, IEEE) received the Diploma degree in applied mathematics from the Technical University of Munich, Munich, Germany, in 2002, and the Ph.D. degree from the Faculty of Electrical Engineering and Information Technology, Technical University of Munich, in 2007.

She is currently a Senior Researcher with the Institute of Automatic Control Engineering, Faculty of Electrical Engineering and Information Technology, Technical University of Munich. Her research interests include optimal control and nonlinear control theory, and applications to robotics.



Martin Buss (Fellow, IEEE) received the Diploma degree in electrical engineering from the Technische Universität Darmstadt, Darmstadt, Germany, in 1990, and the Ph.D. degree in electrical engineering from The University of Tokyo, Tokyo, Japan, in 1994.

In 1988, he was a Research Student with the Science University of Tokyo, Tokyo, for one year. From 1994 to 1995, he was a Post-Doctoral Researcher with the Department of Systems Engineering, Australian National University, Canberra, ACT, Australia. From 1995 to 2000, he was a Senior Research Assistant and a Lecturer with the Chair of Automatic Control Engineering, Department of Electrical Engineering and Information Technology, Technical University of Munich, Munich, Germany. From 2000 to 2003, he was a Full Professor, the Head of the Control Systems Group, and the Deputy Director of the Institute of Energy and Automation Technology, Faculty IV, Electrical Engineering and Computer Science, Technical University Berlin, Berlin, Germany. Since 2003, he has been a Full Professor (Chair) with the Chair of Automatic Control Engineering, Faculty of Electrical Engineering and Information Technology, Technical University of Munich, where he has also been with the Medical Faculty since 2008. Since 2006, he has also been the Coordinator of the Deutsche Forschungsgemeinschaftcluster of excellence—Cognition for Technical Systems, Bonn, Germany. His research interests include automatic control, mechatronics, multimodal human-system interfaces, optimization, nonlinear, and hybrid discrete-continuous systems.