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

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

***Abstract*—For safely applying reinforcement learning algorithms on high-dimensional nonlinear dynamical systems, a simplified system model is used to formulate a safe reinforcement learning framework. Based on the simplified system model, a low-dimensional representation of the safe region is identified and is 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 in this work a general data-driven approach that is able to efficiently learn a low-dimensional representation of the safe region. Through an online adaptation method, the low-dimensional representation is updated by using the feedback data such that more accurate safety estimates are obtained. The performance of the proposed approach for identifying the low-dimensional representation of the safe region is demonstrated with a quadcopter example. The results show that, compared to previous work, a more reliable and representative low-dimensional representation of the safe region is derived, which then extends the applicability of the safe reinforcement learning framework.**

***Index Terms*—safe reinforcement learning, deep learning in robotics and automation, learning and adaptive systems, data-driven model order reduction**

I. Introduction

RECENT studies of applying reinforcement learning or deep reinforcement learning algorithms on 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 exhibit the potential of utilizing reinforcement learning algorithms as a replacement for traditional controller design techniques, most of them are still presented only in simulations [3]. One major impediment of 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, not only the system itself but also the environment may get damaged during the learning. In order to apply state-of-the-art reinforcement learning algorithms on real-world control systems, one central problem to address is introducing a reliable safety guarantee to the learning process.

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

O. Oguz is with the Max Planck Institute for Intelligent Systems and University of Stuttgart (e-mail: ozgur.oguz@ipvs.uni-stuttgart.de).

Providing safety guarantees to reinforcement learning algorithms has been a research topic for over a decade [4]. In earlier studies, usually a manual control mechanism is employed 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 drives the system to a risky state. However, such an approach requires a considerable amount of resource to monitor the entire learning process. Hence in most cases, it is not applicable to complicated learning tasks. Another possibility of safely implementing reinforcement learning algorithms on real-world dynamical systems is given by transfer learning [6]. First, a satisfying initial policy is trained in simulation and then transferred to the real-world dynamical system. In essence, the required number of learning iterations for obtaining the final policy is minimized and thus the risk of encountering dangerous intermediate policy is reduced [7]. However, since the mismatch between simulation and reality is not considered in transfer learning, no reliable safety guarantee is obtained [8].

Recent promising studies on safe reinforcement learning include insight from model-based controller design techniques into the learning process. For example, robust model predictive control methods are introduced to ensure the safety during learning. By learning a model of the system dynamics [9] or of environmental constraints [10], safety criteria given as constraints are satisfied with bounded error. However, here the accuracy of the learned system model strongly affects the performance.

To relax the requirement on the system model, probabilistic safety measures are employed and are evaluated by data-driven methods [11]. E.g. in [12], modelling uncertainties are approximated by Gaussian process models [13] and a safe region is computed through reachability analysis [14]. Similarly in [15], [16], Gaussian process models are used to model unknown system dynamics. Then a safe region is 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. The safety is ensured by switching to a safety controller when the system approaches the boundary of the safe region. However, the computation of the safe region is performed either through solving a partial differential equation in [12] or through sampling in [16], 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 not only making suitable assumptions about the distribution of dynamics, but also acquiring a sufficient amount of data are difficult. Therefore, although approaches like [12], [16] provide promising results for low-dimensional dynamical systems, they are not directly applicable to complex dynamical systems [17].

Often the motivation of using reinforcement learning algorithms for controller design is to overcome the difficulty of applying model-based controller design approaches for highly nonlinear, high-dimensional and uncertain dynamic system models. It is in particular challenging to compute a safe region for a complex dynamical system. Therefore, [18] introduces a safe reinforcement learning (SRL) framework that is based on finding a simplified system. It allows to efficiently compute a simplified safe region as an approximation for the safe region of the full dynamics. Such a low-dimensional representation of the safe region provides at least safety estimates for original system states, and it can be updated during the learning process to have more accurate safety estimates. A probabilistic safety measure is used to represent the uncertainty that comes from making safety decisions for the complex dynamics based on a rough low-dimensional reduction. In [18], a physically inspired model order reduction [19] is used to obtain the simplified system. However, implementing such a technique usually requires a thorough understanding about the system dynamics. Moreover, multiple tests on the performance are required before a satisfying simplified system can be found.

In this work, we focus on a general approach that is able to efficiently learn a low-dimensional representation of the safe region for the SRL framework. Inspired by transfer learning [20], we assume that an approximated system model of the complex dynamical system is available. Although inevitably, the approximated model demonstrates discrepancies compared to the real system behavior, an initial estimate of safety is usually obtainable through simulating the approximated model. For example, while the dynamics of a real-world humanoid cannot be perfectly known, 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 different original system states. However, due to the high-dimensional state space, it is infeasible to acquire a sufficient amount of training data for directly learning the safe region of the original system. To solve this problem, a data-driven method that computes probabilistic similarities between each training data is implemented to learn a low-dimensional representative safety feature of the complex dynamical system. Based on the learned feature, a low-dimensional representation of the safe region is constructed, which is used as the starting point to SRL of the real system. During the learning process, we receive feedback data about the actual safe region of the real system. This is used in the proposed online adaptation method to improve the low-dimensional representation of the safe region such that more accurate safety estimates are acquired.

The contribution of this work is two-fold. First, we introduce a data-driven method that is capable of systematically identifying a low-dimensional representation of the safe region for the SRL framework. Second, we propose an efficient online adaptation method to update the low-dimensional representation according to the real system behavior. We use the same SRL framework as in [18] but the safety decision is made upon different concepts. Compared to [18], our approach results in a more reliable and representative low-dimensional representation of the safe region. It reduces the required effort for finding a simplified system model, which then extends the applicability of the SRL framework.

The remainder of this paper 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. To derive a low-dimensional representation of the safe region, we propose a data-driven method in Section IV, followed by an online adaptation method given in Section V to update the low-dimensional representation. An example is provided in Section VI to demonstrate the performance of the proposed approach. In Section VII, we discuss several properties of the approach. Finally, Section VIII concludes this work.

II. Safe Reinforcement Learning Framework

The purpose of SRL is 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 [18]. 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 state back to a safe state.

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 $\mathcal{X}$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ 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. Through a state transform, any equilibrium point can be shifted to the origin. Therefore, we only utilize the origin to formulate the safe region in this work.

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

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

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

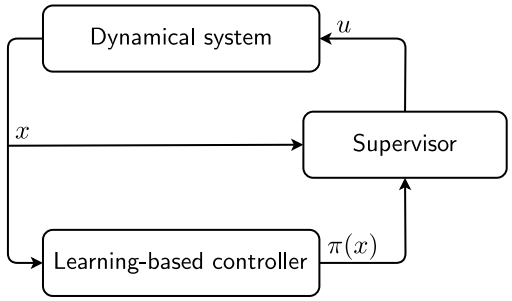


Fig. 1: SRL framework with a supervisor which decides on the actual applied actions.

initial states that can be driven back to a safe state, i.e., the origin, under the control policy $K(x)$. Therefore in this work, 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

The SRL framework 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 that 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 the safety. After this safety recovery, the learning environment is reset and the next learning iteration starts again with time $t = 0$.

Remark 1. *For simplicity, we only consider the safe region obtained from the ROA $\mathcal{R}$ in this work. Other concepts can also be used to construct the safe region, e.g. the maximal control invariant set [21] and invariance functions [22]. The SRL framework and the approach proposed in this work are compatible with these different definitions, as long as they provide a closed and control invariant safe region 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 [23]. To overcome this problem, a SRL framework that is based on estimating the safety with a low-dimensional representation of the safe region is introduced [18].

Each original system state x is mapped to a low-dimensional safety feature, represented as a simplified state $y \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$, 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 through 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 work. On the one hand, the SAF $\Gamma(y)$ encodes information about the safety of the simplified state y. On the other hand, it also includes the uncertainty 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 $y = \Psi(x)$ as well as the SAF $\Gamma(y)$ with 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 that 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 about this SRL framework are given in [18].

III. Overview of the Approach

The essential part of applying the SRL framework on complex dynamical systems is finding a reliable low-dimensional representation of the safe region $\mathcal{S}_y$. To achieve this, a general data-driven approach is proposed in this work.

We consider the scenario where the complex dynamical system, referred to as the real system, has partially unknown dynamics. Nevertheless, 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

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

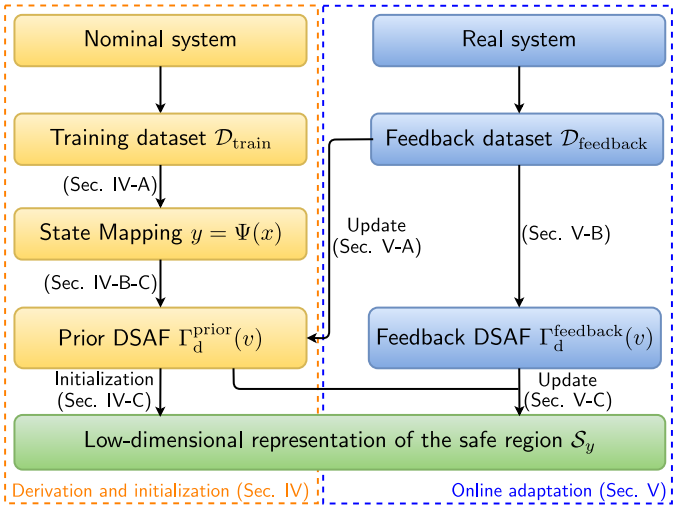


Fig. 2: Overview of the proposed approach. The low-dimensional representation $\mathcal{S}_y$ is initialized by using the training dataset $\mathcal{D}_{\text{train}}$ obtained from the nominal system. Once we collect the feedback dataset $\mathcal{D}_{\text{feedback}}$ on the real system, the low-dimensional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

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

Due to the highly nonlinear and high-dimensional dynamics, the calculation of the safe region is computationally infeasible both for the nominal and the real systems. Besides, although the real system provides exact information about the safety, in general it is expensive to collect data directly on the real system. In contrast, simulating the nominal system is usually possible efficiently and allows to obtain a sufficient amount of data for finding a low-dimensional safety representation. However, due to the unknown term $d(x)$, such data is inaccurate and has 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 an approach that is outlined in Fig. 2. It consists of two parts that solve the following two problems respectively:

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.

Part 1) Derivation and Initialization

Since no information about the 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*

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

It contains the simulation results of whether the safety recovery is successful or not for different system states x under the corrective controller $K(x)$. The i-th training data includes three elements

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

x_{sim}^i is the initial system state where 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 derivation and initialization of the low-dimensional representation of the safe region $\mathcal{S}_y$ is thus performed by using the training dataset $\mathcal{D}_{\text{train}}$. To do this, we first identify the state mapping $y = \Psi(x)$ through a data-driven method that computes the probabilistic similarity between each training data (Section IV-A). Then for having 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 safety assessment function (DSAF)* $\Gamma_{\text{d}}(v)$. 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 $\Gamma_{\text{d}}(v)$ (Section IV-B). To enable the SRL framework on the real system, we also calculate an initial estimate of the DSAF $\Gamma_{\text{d}}(v)$, denoted as the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$, from the training dataset $\mathcal{D}_{\text{train}}$. It is then used as an initialization of the low-dimensional representation of the safe region $\mathcal{S}_y$ (Section IV-C). More details of part 1) are given in Section IV.

Part 2) Online Adaptation

Due to the unknown part of the system dynamics $d(x)$, there exists an inevitable mismatch between simulation and reality. To have more accurate safety estimates, we therefore update the low-dimensional representation $\mathcal{S}_y$ by accounting for the real system behavior.

During the learning, every time the corrective controller $K(x)$ is activated, we observe a 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*

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

It contains the results of safety recovery by implementing the corrective controller $K(x)$ on the real system. The i-th feedback data is

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

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

Due to the fact that 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 update of the low-dimensional representation of the safe region $\mathcal{S}_y$ needs to be performed in a data-efficient way. To achieve this, we propose an online adaptation method in Section V. It is composed of three steps: First, we modify the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ by changing our confidence in its reliability using the feedback dataset $\mathcal{D}_{\text{feedback}}$ (Section V-A). Second, for fully utilizing the valuable information contained in the feedback dataset $\mathcal{D}_{\text{feedback}}$, we generate another feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$ (Section V-B). Third, the two DSAFs are fused to obtain a more accurate DSAF $\Gamma_{\text{d}}(v)$, 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 $y = \Psi(x)$ that gives the low-dimensional safety feature, and the SAF $\Gamma(y)$ that 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) [24], which was originally proposed for visualizing high-dimensional data.

A. Identify 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 $y^1, \dots, y^{k_t}$, that best corresponds to 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 two- or three-dimensional data point in a way 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 the safety rather than just the distance. Accordingly, we propose a new metric that considers similarity and safety at the same time.

The general motivation of 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 the safe region is defined with respect to the ROA in this work, 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 i-th and j-th training data by using dynamic time warping (DTW) [25] as

$$\omega_{ij} = \text{dtw}(\Phi_{\text{sim}}(t; x^i_{\text{sim}}), \Phi_{\text{sim}}(t; x^j_{\text{sim}})) \tag{13}$$

where $\text{dtw}(\cdot)$ 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.

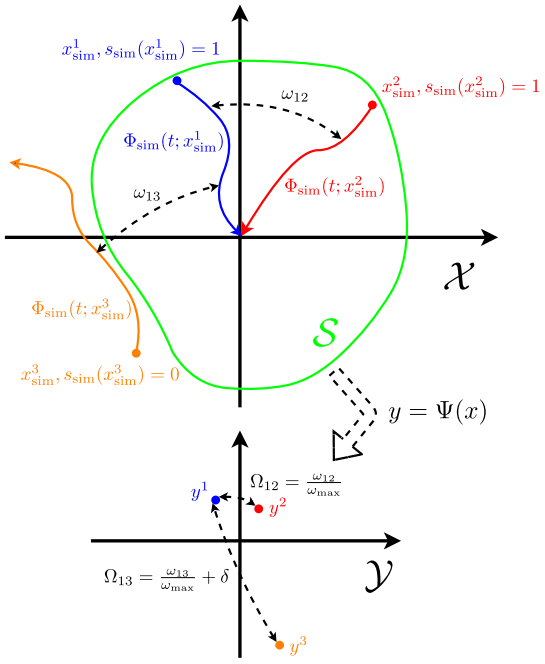


Fig. 3: For three training data D^1_{train}, D^2_{train}, D^3_{train}, the distance Ω_{12} and Ω_{13} are computed by using the trajectory distances ω_{12}, ω_{13} and the safety labels $s_{\text{sim}}(x^1_{\text{sim}})$, $s_{\text{sim}}(x^2_{\text{sim}})$, $s_{\text{sim}}(x^3_{\text{sim}})$. Based on the 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.

Remark 2. *Besides DTW, other trajectory distance measures, e.g., Fréchet distance [26], 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 original system states x^i_{sim} and x^j_{sim} have the same safety property, it is still possible that safe and unsafe states have similar trajectories. For obtaining a better low-dimensional safety feature, we thus modify the trajectory distance ω_{ij} depending on the safety label $s_{\text{sim}}(x_{\text{sim}})$ and compute the distance Ω_{ij} between the i-th and j-th training data as

$$\Omega_{ij} = \begin{cases} \dfrac{\omega_{ij}}{\omega_{\max}} + \delta, & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) \neq s_{\text{sim}}(x^j_{\text{sim}}) \\ \dfrac{\omega_{ij}}{\omega_{\max}}, & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) = s_{\text{sim}}(x^j_{\text{sim}}) \end{cases} \tag{14}$$

where δ is a constant and $\omega_{\max} = \max_{i,j} \omega_{ij}$ is the maximal 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 out that a small value of δ is sufficient for providing a satisfying result of t-SNE, e.g., we use $\delta = 0.01$ in this work. A large value of δ, in contrast, may lead to the ignorance of the information contained in trajectories, which can reduce the representation power of the learned simplified states y.

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. For that, we modify the conditional probability $p_{j|i}$ of t-SNE [24] by using the distance Ω_{ij} as

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

where σ_i is the variance of the Gaussian distribution that is centered on the state x^i_{sim}. The remaining computations are the same as in t-SNE. Since this part delivers no contribution, we only outline important steps of performing t-SNE in Appendix. More details are presented in [24].

By 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 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 two- or three-dimensional, i.e., $y \in \mathbb{R}^{n_y}$ with $n_y = 2$ or $n_y = 3$. In this work, 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 by 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 work, is then utilized to represent the state mapping $y = \Psi(x) = \text{NN}(x)$.

Remark 3. *Different forms of function approximator, e.g., Gaussian process, can be used to describe the state mapping $y = \Psi(x)$. The selection of function approximator depends mainly on the available number of 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 $y = \Psi(x)$ (see the simulations in Section VI-B and in particular Fig. 7 for an example). 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 $y^1, \ldots, y^{k_t}$ again with the learned state mapping. Such a final realization of the low-dimensional safety feature is then used for formulating the SAF $\Gamma(y)$.

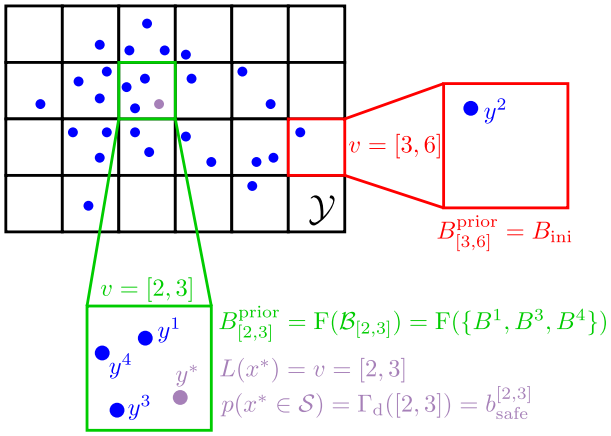


Fig. 4: The simplified state space $\mathcal{Y}$ is discretized into grid cells. The location of each grid cell is indicated by the index vector v. The safety of a new original system state, e.g. x^*, is estimated through the corresponding belief mass as $p(x^* \in \mathcal{S}) = \Gamma_{\text{d}}([2,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

B. Belief Function Theory and DSAF

Once the state mapping $y = \Psi(x)$ is determined, we are able to generate the SAF $\Gamma(y)$ by using the training dataset $\mathcal{D}_{\text{train}}$. However, due to the limited size of 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 decided by the maximal and minimal values of simplified states $y^1, \ldots, y^{k_t}$ in each dimension. Then, we discretize the simplified state space $\mathcal{Y}$ into grid cells with a predefined step size. Each grid cell is assigned with an index vector $v \in \mathbb{Z}_+^2$ to indicate its position in the simplified state space $\mathcal{Y}$, e.g., $v = [2, 3]$ refers to the grid cell that locates at the second row and third column (see Fig. 4). A locating function is defined as follows.

Definition 4. *For an original system state x, by locating its corresponding simplified state $y = \Psi(x)$ in the simplified state space $\mathcal{Y}$, the locating function $L(x)$ returns the index vector v 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 $\Gamma_{\text{d}}(v)$ we will have to define. Then instead of using the simplified state y, the safety of an original system state x is estimated through the index vector v as

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

In general, the DSAF $\Gamma_{\text{d}}(v)$ for an index vector v can be approximated by the number of safe and unsafe original system states x that map to the corresponding grid cell, i.e., $L(x) = v$. However, due to the high-dimensional original system state space, in most cases it is infeasible to acquire

a sufficient amount of data for deriving an accurate estimate. To solve this problem, we propose to use belief function theory [27] to describe the DSAF $\Gamma_{\mathrm{d}}(v)$, where the uncertainty caused by insufficiency in data amount is considered by a subjective probability [28].

Belief function theory is a general approach to model epistemic uncertainty by using 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 [29]. According to this, we define a BBA B_v separately for each index vector v as follows.

Definition 5. *The BBA B_v for an index vector v is given as*

$$B_v = (b^v_{\mathrm{safe}}, b^v_{\mathrm{unsafe}}, \mu^v) \tag{17}$$

which represents the belief about the value of the DSAF $\Gamma_{\mathrm{d}}(v)$ for the index vector v. The belief masses b^v_{safe} and b^v_{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 v from the locating function $L(x)$. μ^v is the subjective uncertainty that reflects the confidence level of estimating the safety. $\mu^v = 0$ means we believe that the estimate is absolutely correct. It holds that

$$b^v_{\mathrm{safe}} + b^v_{\mathrm{unsafe}} + \mu^v = 1 \tag{18}$$

and b^v_{safe}, b^v_{unsafe}, μ^v all lie within the interval $[0, 1]$.

Hence the DSAF $\Gamma_{\mathrm{d}}(v)$ is given by the belief masses b^v_{safe} of the corresponding BBAs B_v as

$$\Gamma_{\mathrm{d}}(v) = b^v_{\mathrm{safe}}. \tag{19}$$

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

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

where p_t is the predefined probability threshold. To enable the application of the SRL framework on the real system, we explain in the next subsection how to initialize the DSAF $\Gamma_{\mathrm{d}}(v)$.

C. Initialize the DSAF from Training Data

Since every training data provides a piece of information about the value of the DSAF $\Gamma_{\mathrm{d}}(v)$, the low-dimensional representation of the safe region $\mathcal{S}_y$ is initialized using the training dataset $\mathcal{D}_{\mathrm{train}}$. 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 $\Gamma_{\mathrm{d}}(v)$.

Definition 6. *The BBA B^i obtained from the i-th training data D^i_{train} is defined as*

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

It represents the belief about the value of the DSAF $\Gamma_{\mathrm{d}}(v)$ for the index vector $v = L(x^i_{\mathrm{sim}})$, where the belief source is the i-th 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 $\mu_{\mathrm{ini}} > 0$ as

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

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

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

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 $\Gamma_{\mathrm{d}}(v)$ 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

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

where B_{ini} is an initial estimate that represents our guess about the BBA B_v when no training data is available (see Fig. 4). $\mathrm{F}(\cdot)$ is a fusion operation among the set $\mathcal{B}_v$ that is referred to as weighted belief fusion and is defined according to [30] as

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

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

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

We refer to this estimate of the BBA B_v as the prior estimate B_v^{prior}. 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 minimal number $k_{\min}$. Otherwise, the prior estimate B_v^{prior} 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 few experience in form of training data for one grid cell, then the respective states will initially be considered as unsafe. The resulting prior estimate B_v^{prior} is a BBA that satisfies

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

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

After computing the prior estimate B_v^{prior} for all index vectors v, we thus obtain a prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$

$$\Gamma_{\text{d}}^{\text{prior}}(v) = b_{\text{safe}}^{v,\text{prior}} \tag{29}$$

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

V. Safety Assessment Function Online Adaptation

In early learning phase with the real system, the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ allows for a rough estimate on safety of an original system state. During the learning process, the feedback data is used to update the DSAF $\Gamma_{\text{d}}(v)$ for achieving more accurate safety estimates. Each update iteration of the DSAF $\Gamma_{\text{d}}(v)$ consists of three steps. First, we modify the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ by revising the subjective uncertainties of the BBAs of the training data. Second, we compute a feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$ by using the feedback data. Third, the updated DSAF $\Gamma_{\text{d}}(v)$ is obtained by fusing the prior and the 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 $\mathcal{D}_{\text{feedback}}$ is incrementally increased during the learning process. For simplicity, we consider the feedback dataset $\mathcal{D}_{\text{feedback}}$ with size k_f in this section. Details of the online adaptation method are given as follows.

A. Update of the Prior DSAF with Feedback Data

The prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ is constructed by using the training dataset $\mathcal{D}_{\text{train}}$, where the uncertainty caused by the unknown term $d(x)$ is represented by the subjective uncertainty μ^i of each BBA B^i. Hence now the update of the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ will modify the subjective uncertainties by accounting for new information given by feedback data. For this, we assume that original system states that are close 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 $||x^1 - x^2||$.*

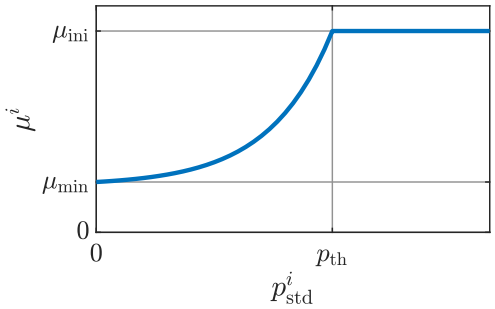


Fig. 5: The subjective uncertainty μ^i of the i-th training data is updated by using the standard deviation p_{std}^i obtained from the GPR model $\text{GP}(x)$.

In addition, we define a function $P(x)$ to quantify the similarity with respect to 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 $\{P(x_{\text{real}}^1), \ldots, P(x_{\text{real}}^{k_f})\}$ for the function $P(x)$, where we have

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

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

Remark 4. *If the real system is a real-world dynamical system, then usually it is 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 feedback data, which then provides the possibility 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 i-th training data, we compute a predicted mean value of the

function $P(x^i_{\mathrm{sim}})$, denoted as p^i_{mean}, from the GPR model $\mathrm{GP}(x)$, along with a corresponding standard deviation p^i_{std} of the predicted value. Since a low value of the standard deviation p^i_{std} means we have observed enough feedback data to make a reliable prediction, we only update the BBA B^i if the standard deviation p^i_{std} is smaller than a predefined threshold p_{th}

$$B^i = \begin{cases} (p^i_{\mathrm{mean}}(1-\mu^i), (1-p^i_{\mathrm{mean}})(1-\mu^i), \mu^i), \\ \qquad \text{if } p^i_{\mathrm{std}} \leq p_{\mathrm{th}} \text{ and } s_{\mathrm{sim}}(x^i_{\mathrm{sim}}) = 1 \\ ((1-p^i_{\mathrm{mean}})(1-\mu^i), p^i_{\mathrm{mean}}(1-\mu^i), \mu^i), \\ \qquad \text{if } p^i_{\mathrm{std}} \leq p_{\mathrm{th}} \text{ and } s_{\mathrm{sim}}(x^i_{\mathrm{sim}}) = 0 \end{cases} \tag{32}$$

with the new subjective uncertainty μ^i calculated as

$$\mu^i = \frac{\mu_{\mathrm{ini}} - \mu_{\min}}{\alpha^{p_{\mathrm{th}}} - 1}(\alpha^{p^i_{\mathrm{std}}} - 1) + \mu_{\min} \tag{33}$$

where μ_{ini} is the same initial subjective uncertainty given in (22). BBAs B^i with $p^i_{\mathrm{std}} > p_{\mathrm{th}}$ remain unchanged, see (22).

Such an update of the BBA B^i considers the predicted value of the function $P(x^i_{\mathrm{sim}})$ and the reliability of this prediction at the same time. (33) is designed by considering two aspects: first, the subjective uncertainty μ^i is set equal to μ_{ini} when $p^i_{\mathrm{std}} = p_{\mathrm{th}}$; second, due to the inevitable reality gap, the subjective uncertainty μ^i maintains a minimal uncertainty $\mu_{\min}$ even when the standard deviation p^i_{std} is 0 (see Fig. 5). We use the exponential form such that the decrease in μ^i is faster when the standard deviation p^i_{std} 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^i_{std} and the threshold p_{th} might change during the learning process. For example, we may obtain $p^i_{\mathrm{std}} \leq p_{\mathrm{th}}$ in the current update iteration, but in the next update iteration it changes to $p^i_{\mathrm{std}} > p_{\mathrm{th}}$. This happens mostly when we first observe a safe original system state but followed by a nearby unsafe state, then the safety of the states that are 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 are updated with the up-to-date feedback dataset $\mathcal{D}_{\mathrm{feedback}}$, the prior estimate B^{prior}_v for each index vector v is recomputed according to (24). This results in an updated prior DSAF $\Gamma^{\mathrm{prior}}_{\mathrm{d}}(v)$, which is used later for revising the DSAF $\Gamma_{\mathrm{d}}(v)$.

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 $\Gamma^{\mathrm{feedback}}_{\mathrm{d}}(v)$, by using the feedback dataset $\mathcal{D}_{\mathrm{feedback}}$.

Due to the insufficient data amount, we consider the estimate obtained from the feedback data also as a subjective probability [18]. Then similar to the prior estimate B^{prior}_v, we formulate another estimate of the BBA B_v for each index vector v as

$$B^{\mathrm{feedback}}_v = (b^{v,\mathrm{feedback}}_{\mathrm{safe}}, b^{v,\mathrm{feedback}}_{\mathrm{unsafe}}, \mu^{v,\mathrm{feedback}}) \tag{34}$$

which is referred to as the feedback estimate B^{feedback}_v.

For each index vector v, the feedback estimate B^{feedback}_v is determined through the number of safe and unsafe feedback data that corresponds to this grid cell. By sorting the feedback dataset $\mathcal{D}_{\mathrm{feedback}}$ with the locating function $L(x)$, we denote the number of safe feedback data that has the index vector v from the locating function, i.e., $L(x_{\mathrm{real}}) = v$ and $s_{\mathrm{real}}(x_{\mathrm{real}}) = 1$, as k^v_{safe} (and k^v_{unsafe} for the number of unsafe feedback data). If at least one feedback data is available for the index vector v, i.e., $k^v_{\mathrm{safe}} + k^v_{\mathrm{unsafe}} \geq 1$, we compute the feedback estimate B^{feedback}_v as follows

$$b^{v,\mathrm{feedback}}_{\mathrm{safe}} = \frac{k^v_{\mathrm{safe}}}{k^v_{\mathrm{safe}} + k^v_{\mathrm{unsafe}}}(1 - \mu^{v,\mathrm{feedback}}) \tag{35}$$

$$b^{v,\mathrm{feedback}}_{\mathrm{unsafe}} = \frac{k^v_{\mathrm{unsafe}}}{k^v_{\mathrm{safe}} + k^v_{\mathrm{unsafe}}}(1 - \mu^{v,\mathrm{feedback}}) \tag{36}$$

$$\mu^{v,\mathrm{feedback}} = \beta \exp(-\gamma(k^v_{\mathrm{safe}} + k^v_{\mathrm{unsafe}} - 1)). \tag{37}$$

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

By using the feedback estimate B^{feedback}_v, we thus get the following feedback DSAF $\Gamma^{\mathrm{feedback}}_{\mathrm{d}}(v)$

$$\Gamma^{\mathrm{feedback}}_{\mathrm{d}}(v) = b^{v,\mathrm{feedback}}_{\mathrm{safe}} \tag{38}$$

which represents the estimate of the DSAF $\Gamma_{\mathrm{d}}(v)$ derived only from the feedback data. In the next subsection, we fuse the feedback DSAF $\Gamma^{\mathrm{feedback}}_{\mathrm{d}}(v)$ with the updated prior DSAF $\Gamma^{\mathrm{prior}}_{\mathrm{d}}(v)$ to derive a more accurate DSAF $\Gamma_{\mathrm{d}}(v)$.

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 $\Gamma_{\mathrm{d}}(v)$, we fuse these two functions via weighted belief fusion as given in (25-27). This leads to a fused estimate B^{fuse}_v for each index vector v

$$B^{\mathrm{fuse}}_v = (b^{v,\mathrm{fuse}}_{\mathrm{safe}}, b^{v,\mathrm{fuse}}_{\mathrm{unsafe}}, \mu^{v,\mathrm{fuse}}) \tag{39}$$

which is computed as

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

If the feedback estimate B^{feedback}_v is non-empty, we find the fused estimate B^{fuse}_v through weighted belief fusion $\mathrm{F}(\cdot)$ of the set $\{B^{\mathrm{prior}}_v, B^{\mathrm{feedback}}_v\}$. Otherwise, we set the fused estimate B^{fuse}_v equal to the prior estimate B^{prior}_v.

The fused estimate B^{fuse}_v fulfills the following property that is also given in [18].

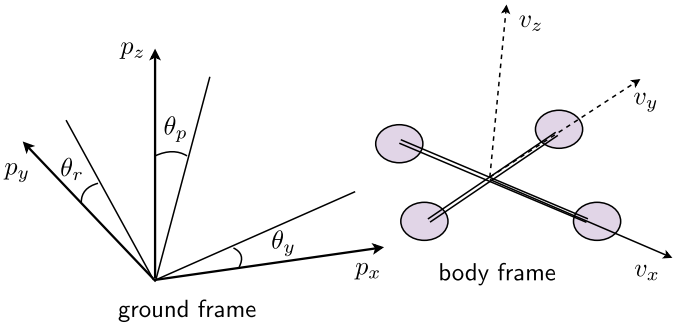


Fig. 6: The system state x of a quadcopter is defined by using the ground frame and the body frame.

Proposition 1. *If the number of feedback data approaches infinity, the fused estimate B_v^{fuse} becomes the actual probabilities and the prior estimate B_v^{prior} has no effect in making safety estimates.*

Proof. Proposition 1 is justified by the following equations

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

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

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

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

Considering the computational efficiency, in general the update of the DSAF $\Gamma_{\text{d}}(v)$ is 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 $\Gamma_{\text{d}}^{\text{prior}}(v)$ and to construct the feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$. Then the fused estimate B_v^{fuse} is computed from these two functions for each index vector v. The updated DSAF $\Gamma_{\text{d}}(v)$ is thus obtained by using the fused estimate B_v^{fuse} as

$$\Gamma_{\text{d}}(v) = b_{\text{safe}}^{v,\text{fuse}} \tag{44}$$

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

VI. Quadcopter Experiments

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

A. Experimental Setup

We simulate the quadcopter by using the system dynamics given in [31]. The 12-dimensional system state is defined as $x = [p_g, \theta_g, v_b, \omega_b]^T$, 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, $v_b = [v_x, v_y, v_z]^T$ and $\omega_b = [\omega_r, \omega_p, \omega_y]^T$ are the linear and angular velocities defined in the body frame (see Fig. 6). 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 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, the corrective controller $K(x)$ is a PID controller [31].

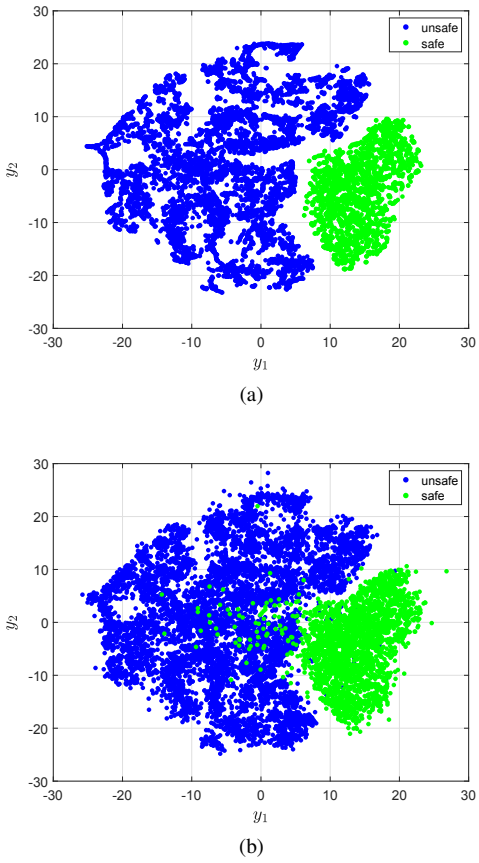


Fig. 7: (a) The initial realization of simplified states $y^1, \ldots, y^{k_t}$ obtained from t-SNE. The safe and unsafe training data are denoted by green and blue points, respectively. (b) The final realization of simplified states $y^1, \ldots, y^{k_t}$ obtained by recomputing with the learned neural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

To generate the training dataset $\mathcal{D}_{\text{train}}$, we first create $k_t = 10000$ original system states x. We set $p_x = p_y = 0$ and $p_z = 2$ m for leaving enough space and time to the corrective controller $K(x)$. All other variables are sampled with a uniform distribution within the following range: $0 \le \theta_r, \theta_p, \theta_y \le 2\pi$ rad, -3 m/s $\le v_x, v_y, v_z \le 3$ m/s, -10 rad/s $\le \omega_r, \omega_p, \omega_y \le 10$ rad/s. Then, the training dataset $\mathcal{D}_{\text{train}}$ is obtained by examining the performance of the corrective controller $K(x)$ for all these initial values.

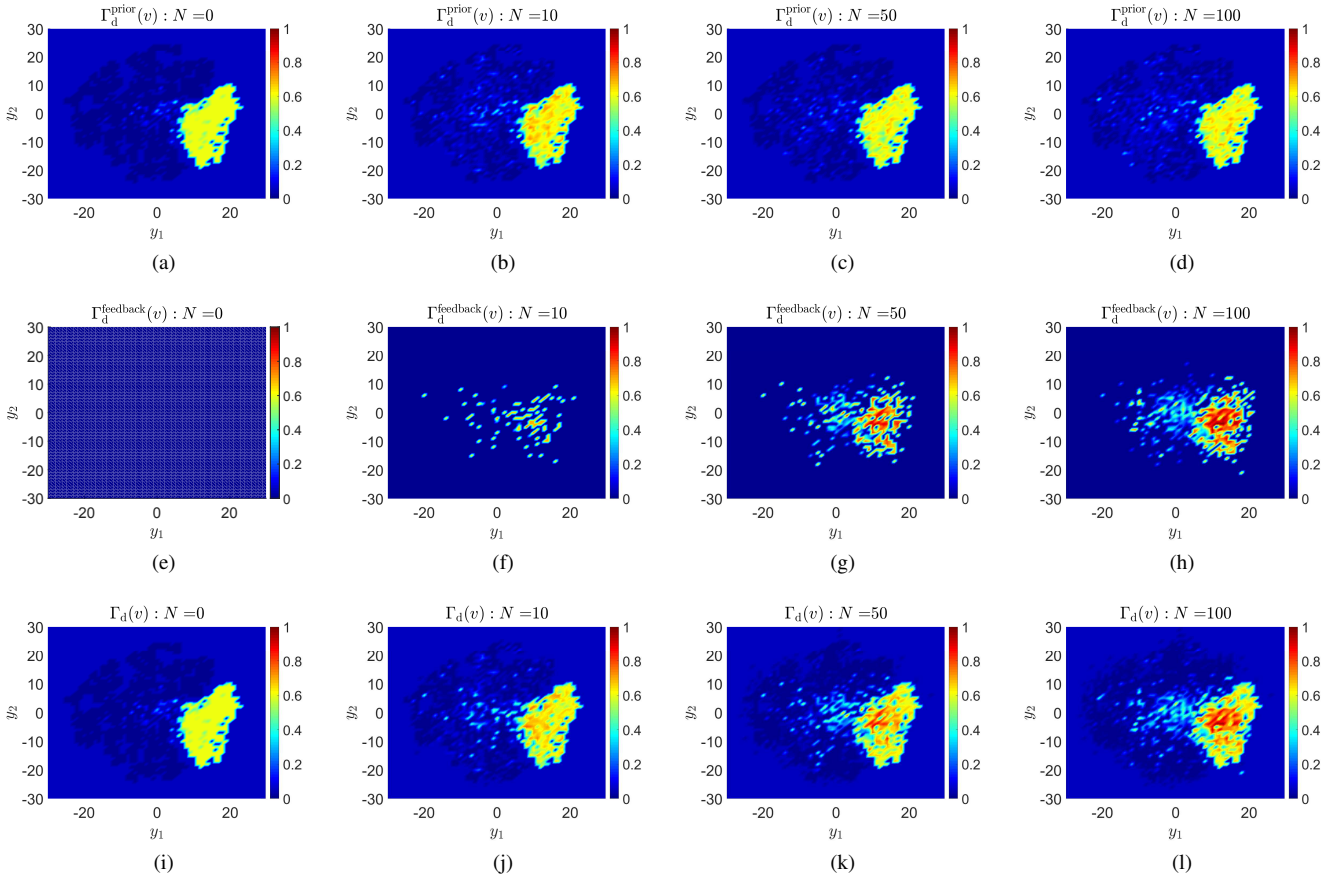


Fig. 8: Results of the online adaptation. (a)-(d) The prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ in different update iterations N. $N = 0$ refers to the initialization prior to the online adaptation. The values are represented by different colors. (e)-(h) The feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ in different update iterations N. (i)-(l) The DSAF $\Gamma_{\mathrm{d}}(v)$ in different update iterations N.

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

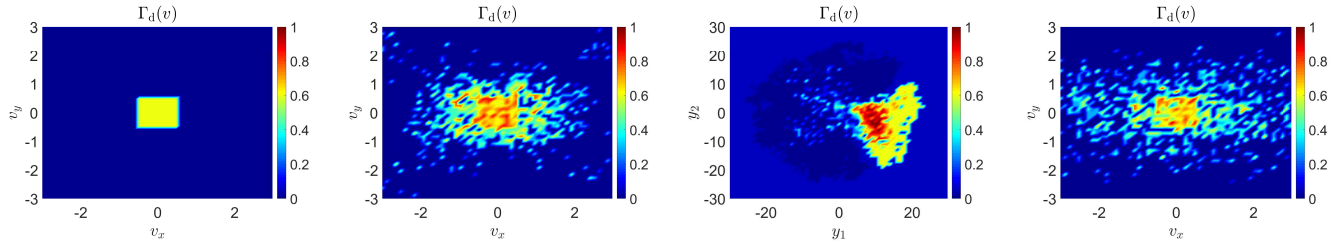
The initial realization of the low-dimensional safety feature, i.e., the values of simplified states $y^1, \ldots, y^{k_t}$, obtained from t-SNE is given in Fig. 7a. We use $\delta = 0.01$ in (14) and set the perplexity and the tolerance of t-SNE (see [24]) to 40 and $1e^{-4}$, respectively. The result shows that, the safe and unsafe original system states are clearly separated in the two-dimensional 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 by using the initial realization of simplified states $y^1, \ldots, y^{k_t}$ and the set of original system states $\{x_{\mathrm{sim}}^1, \ldots, x_{\mathrm{sim}}^{k_t}\}$. Through recomputing the outputs of the learned neural network, we obtain the final realization of the low-dimensional safety feature, i.e., the values of simplified states $y^1, \ldots, y^{k_t}$, that is given in Fig. 7b. Due to the approximation error, certain simplified states have a slightly changed position compared to the values obtained from t-SNE. Nevertheless, this does not affect the computation of the low-dimensional representation of the safe region $\mathcal{S}_y$, as later in the online adaptation the results are updated by 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 $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ is thus computed from the training dataset $\mathcal{D}_{\mathrm{train}}$ by using the index vector v. The results are given in Fig. 8a, where the initial subjective uncertainty, the initial estimate and the minimal number are selected as $\mu_{\mathrm{ini}} = 0.4$, $B_{\mathrm{ini}} = (0.05, 0.55, 0.4)$ and $k_{\min} = 3$, respectively. Depending on the number of safe and unsafe training data in each grid cell, the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ estimates the probability $p(x \in \mathcal{S})$ for original system states x that have the index vector v from the locating function $L(x)$. In Fig. 8i, the DSAF $\Gamma_{\mathrm{d}}(v)$ is initialized by the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$. In the next subsection, we demonstrate the update process of the DSAF $\Gamma_{\mathrm{d}}(v)$ by employing the proposed online adaptation method.

C. Updating the Low-dimensional Representation

For simulating a mismatch between the nominal and the real systems, we set the mass and the maximal lifting force of the real system to $m = 0.8$ kg and $f = 145$ N, respectively. To eliminate the influence of specific learning task or algorithm and focus on illustrating the update process, the feedback dataset $\mathcal{D}_{\mathrm{feedback}}$ is obtained by randomly selecting states x_{real}



(a) (b) (c) (d)

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

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: $\mu_{\min} = 0.1$, $p_{\mathrm{th}} = 0.3$, $\alpha = 3e^5$, $\beta = 0.3$, $\gamma = 0.4$. The GPR model $\mathrm{GP}(x)$ 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 $\mathcal{D}_{\mathrm{feedback}}$. The DSAF $\Gamma_{\mathrm{d}}(v)$ 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 $\Gamma_{\mathrm{d}}(v)$ is initialized as the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$, while the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ is constructed by using the empty BBA $B_{\varnothing}$ (see Fig. 8a, 8e, 8i). Once the learning starts, we collect the feedback data incrementally. In the early updating phase, e.g., update iteration $N = 10$, the DSAF $\Gamma_{\mathrm{d}}(v)$ is mainly determined by the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$. The subjective uncertainties of each training data are modified by using the feedback data, where we become confident about the safety of certain training data if we observe a nearby feedback data that has the same safety property as it. Since the amount of feedback data is insufficient for providing a reliable safety estimate, the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ has a lesser effect on computing the low-dimensional representation of the safe region $\mathcal{S}_y$ (see Fig. 8b, 8f, 8j).

When more feedback data are available, e.g., update iteration $N = 50$, the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ is able to provide more accurate safety estimates, hence its influence on the DSAF $\Gamma_{\mathrm{d}}(v)$ also becomes more significant. Due to the high dimensionality of the original system state x and the limited number of feedback data, it is difficult to acquire an estimate with a high confidence from the GPR model $\mathrm{GP}(x)$. As a result, changes are marginal in the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ (see Fig. 8c, 8g, 8k). With even more feedback data, e.g., update iteration $N = 100$, the DSAF $\Gamma_{\mathrm{d}}(v)$ 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 $\Gamma_{\mathrm{d}}(v)$ 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. 8d, 8h, 8l).

D. Comparison with Physically Inspired Model Order Reduction

We compare the proposed approach with the physically inspired model order reduction presented in [18] in terms of representation power of the identified low-dimensional representation of the safe region $\mathcal{S}_y$, i.e., how well the safe and unsafe states are separated. For that, we compute another DSAF $\Gamma_{\mathrm{d}}(v)$ by using physical features. Similar as in [18], the low-dimensional safety feature, i.e., the simplified state y, is selected as the velocities in x and y directions $y = [v_x, v_y]^T$. To avoid dangerous behaviour in early learning phase, the low-dimensional representation of the safe region $\mathcal{S}_y$ is initialized conservatively [18], where we set $\Gamma_{\mathrm{d}}(v) = 0.6$ for grid cells that satisfy $-0.5 \leq v_x, v_y \leq 0.5$ (see Fig. 9a).

As the learning task in [18] 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 $\mathcal{D}'_{\mathrm{feedback}}$ that has the same size as the dataset $\mathcal{D}_{\mathrm{feedback}}$. However, instead of the complete original system state space given in Section VI-A, the states x_{real} in the set $\mathcal{D}'_{\mathrm{feedback}}$ are sampled from a smaller state space, where the ranges of angular positions and angular velocities are changed to $-\frac{1}{3}\pi \leq \theta_r, \theta_p, \theta_y \leq \frac{1}{3}\pi$ rad and -3 rad/s $\leq \omega_r, \omega_p, \omega_y \leq 3$ rad/s, respectively.

We first compare the performance of both approaches by considering a small state space, i.e., the feedback dataset $\mathcal{D}'_{\mathrm{feedback}}$ is used for the update. The result shows 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. 9b). Meanwhile, the proposed approach also presents a satisfying result with a marginally better separation between safe and unsafe states (see Fig. 9c).

However, if the learning task becomes more complicated, usually the complete state space has to be explored for finding an optimal policy. To simulate this scenario, we also update the initial DSAF $\Gamma_{\mathrm{d}}(v)$ by using the feedback dataset $\mathcal{D}_{\mathrm{feedback}}$. As seen in Fig. 9d, 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 exist 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 original system states x more precisely, a satisfying separation between safe and unsafe regions is achieved (see Fig. 8l). Hence, more useful safety estimates are obtained. The independence on 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. Connection to Different SRL Tasks

In [18], the SRL framework works with the low-dimensional representation of the safe region $\mathcal{S}_y$ found by physically inspired model order reduction. The estimated safe region is initialized conservatively and then increased through feedback data. Such a representation is useful when a satisfying policy can be found without requiring an extensive exploration in the original state space. This is often the case if the learning task is relatively simple, e.g. teaching the quadcopter to fly forward in [18], such that reliable safety estimates can be made through physical features. However, when the learning algorithm is supposed to explore a large portion of the state space to find an optimal policy, at least a rough safety assessment is needed for the complete state space. Unfortunately, restricted by the representation power, the physically inspired low-dimensional representation of the safe region $\mathcal{S}_y$ fails to provide satisfying safety estimates in this case.

In contrast, the proposed approach in this work is able to identify a low-dimensional representation of the safe region $\mathcal{S}_y$ that makes more precise predictions about safety. Meaningful safety estimates are obtained even 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 complicated 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 to clearly separate safe and unsafe states for large portions of a high-dimensional state space, see also Section VI-D. Second, the required effort for identifying the low-dimensional representation of the safe region $\mathcal{S}_y$ is low. While e.g. physically inspired model order reduction usually needs a comprehensive analysis of the system dynamics, the proposed approach only relies 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 via the usage of two DSAFs. Hence, the update can be performed with few feedback data while providing a satisfying result.

However, the performance of the proposed approach is limited by the quality of the nominal system model. Since the low-dimensional safety feature is determined only by using the training data, the mismatch may lead to a less meaningful result. In general, for obtaining a good representation power of the low-dimensional representation of the safe region $\mathcal{S}_y$, the uncertainty in the system dynamics $d(x)$ has to be bounded and small. To further generalize the presented approach, more studies are required for quantifying the influence of the unknown term $d(x)$ on the reliability of the obtained safety estimates.

VIII. Conclusion

In this work, we propose a general approach for identifying a low-dimensional representation of the safe region for safe reinforcement learning frameworks. By using training data obtained from a nominal system model, a data-driven method is proposed to formulate and initialize a low-dimensional representation of the safe region. Then according to feedback data collected on the real system, such a low-dimensional representation is updated through an online adaptation method for having more accurate safety estimates. As illustrated by a quadcopter example, the proposed approach is able to efficiently find a reliable and representative low-dimensional representation of the safe region. Since currently the parameters are selected empirically, one possible direction for future work is to find a reliable and efficient way of tuning the parameters. We believe that the presented approach is applicable to a wide range of dynamical systems and learning tasks, and it gives an insight about how to safely employ learning algorithms on real-world scenarios.

Appendix
Computations of t-SNE

To derive a realization of the low-dimensional safety feature that corresponds to the training dataset $\mathcal{D}_{\text{train}}$, we first compute the conditional probability $p_{j|i}$ as given in (15). It stands for the probability that the state x^i_{sim} picks the state x^j_{sim} as its neighbor according to their probability density under a Gaussian distribution. To alleviate the identification problem caused by outliers, the similarity between two training data points is then defined by the joint probability p_{ij}

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2k_t} \tag{45}$$

where k_t is the size of the training dataset $\mathcal{D}_{\text{train}}$. Since we are only interested in the pairwise similarities, we set $p_{ij} = 0$ if $i = j$.

Thereafter, we compute the values of simplified states $y^1, \dots, y^{k_t}$ that best represent the similarity p_{ij}. This is achieved through a similar joint probability q_{ij} of two simplified states y^i and y^j

$$q_{ij} = \frac{(1 + ||y^i - y^j||^2)^{-1}}{\sum_{k \neq i} (1 + ||y^i - y^k||^2)^{-1}} \tag{46}$$

where $||\cdot||$ is the Euclidean distance and we have $q_{ij} = 0$ if $i = j$. A heavy-tailed Student t-distribution is used here to measure the similarity. The values of simplified states $y^1, \ldots, y^{k_t}$ are determined by minimizing a cost function C given as

$$C = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{47}$$

where $\mathrm{KL}(\cdot)$ is the Kullback-Leibler divergence. P and Q are the joint probability distributions in the high-dimensional and low-dimensional state spaces, respectively. The cost function C represents how well the identified simplified states can reproduce the similarities between different training data. Details about the method for solving this optimization problem, as well as the parameter selection, e.g., the variance σ_i in (15), are given in [24].

REFERENCES

[1] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 41, 2017.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[3] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[4] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems*, 2007, pp. 1–8.

[6] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *International Conference on Learning Representations Workshop*, 2017.

[9] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based nmpc enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[10] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Robotics: Science and Systems*, 2016.

[11] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," in *International Conference on Machine Learning*, 2012, pp. 1451–1458.

[12] 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 Transactions on Automatic Control*, 2018.

[13] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[14] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *Conference on Decision and Control*, 2017, pp. 2242–2253.

[15] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *Conference on Decision and Control*, 2016, pp. 4661–4666.

[16] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, 2017, pp. 908–918.

[17] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 8550–8556.

[18] 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 Transactions on Robotics*, 2020.

[19] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model order reduction: theory, research aspects and applications*. Springer, 2008, vol. 13.

[20] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," in *International Conference on Robotics and Automation*. IEEE, 2017, pp. 1557–1563.

[21] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[22] M. Sobotka, J. Wolff, and M. Buss, "Invariance controlled balance of legged robots," in *European Control Conference*, 2007, pp. 3179–3186.

[23] A. A. Ahmadi and A. Majumdar, "Dsos and sdsos optimization: more tractable alternatives to sum of squares and semidefinite optimization," *SIAM Journal on Applied Algebra and Geometry*, vol. 3, no. 2, pp. 193–230, 2019.

[24] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[25] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[26] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Citeseer, Tech. Rep., 1994.

[27] G. Shafer, *A mathematical theory of evidence*. Princeton University Press, 1976, vol. 42.

[28] D. Kahneman and A. Tversky, "Subjective probability: A judgment of representativeness," *Cognitive psychology*, vol. 3, no. 3, pp. 430–454, 1972.

[29] A. Jøsang, *Subjective logic*. Springer, 2016.

[30] ——, "Categories of belief fusion," *Journal of Advances in Information Fusion*, vol. 20, 2018.

[31] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, 2011.