A General Framework to Increase Safety of Learning Algorithms for Dynamical Systems Based on Region of Attraction Estimation

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

***Abstract*—Although the state-of-the-art learning approaches exhibit impressive results for dynamical systems, only a few applications on real physical systems have been presented. One major impediment is that the intermediate policy during the training procedure may result in behaviors that are not only harmful to the system itself but also to the environment. In essence, imposing safety guarantees for learning algorithms is vital for autonomous systems acting in the real world. In this article, we propose a computationally effective and general safe learning framework, specifically for complex dynamical systems. With a proper definition of the safe region, a supervisory control strategy, which switches the actions applied on the system between the learning-based controller and a predefined corrective controller, is given. A simplified system facilitates the estimation of the safe region for the high-dimensional dynamical system. During the learning phase, the belief of the safe region is updated with the actual execution results of the corrective controller, which in turn enables the learning-based controller to have more freedom in choosing its actions. Two examples are given to demonstrate the performance of the proposed framework, one simple inverted pendulum to illustrate the online adaptation method, and one quadcopter control task to show the overall performance.**

***Index Terms*—Deep learning in robotics and automation, learning and adaptive systems, robot safety, safe reinforcement learning.**

I. Introduction

REINFORCEMENT learning and deep reinforcement learning have exhibited attractive results in various tasks, e.g., decision making [1], image processing [2], or natural language processing [3]. This encourages the extension of these algorithms to highly nonlinear and high-dimensional dynamical systems (referred to as complex dynamical systems in this work), where purely model-based controller design might be difficult or even infeasible [4]. Recently developed algorithms, where deep neural networks are utilized as the learning-based controller, have illustrated impressive achievements in these scenarios, such as humanoid control [5] or control of manipulation tasks [6]. However, most of the state-of-the-art results are still presented only in the simulated environment. A major reason is that the inherent random exploration mechanism of many learning algorithms hinders their implementations on real-world problems, as the intermediate policies may lead to harmful behaviors to the system or to the environment. For example, an autonomous mobile robot may collide with obstacles many times before it learns a successful policy. Hence, one central aspect of applying learning algorithms to physical dynamical systems is to effectively impose safety guarantees during the exploration process.

In earlier studies of employing a learning-based controller for real dynamical systems, safety was usually achieved by introducing an additional manual control mechanism, e.g., in [7] and [8], a human pilot takes over the control of the helicopter in case of failure. However, application of such a strategy is quite limited, as monitoring the entire training procedure requires a considerable amount of resources. Later work on transfer learning inspires another possibility of implementing learning techniques on physical systems [9]. A satisfying initial policy is first trained in the simulator and then transferred to the real system [10]. Although this might reduce the total required training time, no safety guarantee has been proposed for the transfer process. Due to the inevitable mismatch between the simulation and the reality, there is still a high probability of encountering a risky intermediate policy during the early learning phase on the real systems [11]. Clearly, a learning approach with a reliable safety guarantee is desirable, as it can expand the range of real-world implementations of modern learning methods on physical systems.

Safety in reinforcement learning has remained an open research problem for over a decade [12]. In discrete action space problems, such as Markov decision processes (MDPs), safety is introduced as an extra component in the reward function. For example, Geibel and Wysotzki [13] incorporated the risk of driving the system to an error state into the expected return while robust MDPs maximize the reward under uncertain transition probabilities [14]. Moldovan and Abbeel [15] also proposed a safe exploration algorithm for MDPs where a constraint about being able to go back to the initial state is imposed. The probabilistic formulation of safety guarantee in [15] inspired later work of using Bayesian optimization as a tool for the safety

Manuscript received November 7, 2019; accepted May 2, 2020. This article was recommended for publication by Associate Editor S. Calinon and Editor A. Billard upon evaluation of the reviewers' comments. *(Corresponding author: Zhehua Zhou.)*

The authors are with the Chair of Automatic Control Engineering, Technical University of Munich, 80290 Munich, Germany (e-mail: zhehua.zhou@tum.de; o.oguz@tum.de; marion.leibold@tum.de; mb@tum.de).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2020.2992981

analysis [16]. An accurate probabilistic system model is needed for these aforementioned approaches. For tasks with continuous action space, the safety has been tackled as a constraint satisfaction problem in a model-free learning algorithm [17]. Bayesian optimization and Gaussian process models have been employed to give high-probability worst-case safety guarantees. However, such a strategy usually results in a task-specific policy and the state constraints may still be violated as no actual system model is considered.

For scenarios where a system model is available, recent studies provide reliable safety analysis by incorporating control-theoretical concepts into the learning algorithms. In [18], a Lyapunov-based reinforcement learning approach that switches between several predefined safe controllers is proposed, however, finding these prior controllers might be a challenging task. Robust model-predictive control was applied to impose safety guarantees on the learning process, e.g., by learning a model of the uncertain environmental constraints [19] or system dynamics [20], robust feasibility and constraint satisfaction are guaranteed with bounded error [21]. However, the performance of such robust controllers depends on the accuracy of the learned model. Recent research has presented more promising results by combining data-driven and model-based techniques. In [22], the safe exploration during learning is modeled as a differential game, where the controller keeps the system within a safe region under external disturbances. The safe region is obtained by reachability analysis where the optimal solution is calculated from the Hamilton–Jacobi–Isaacs equation [23]. The uncertainty is modeled as a Gaussian process, which is updated during the learning procedure. Similarly, Berkenkamp *et al*., [24] suggested an approach where the exploration of the learning algorithm is limited to the region of attraction (ROA). A prior system model with partially unknown dynamics is utilized to safely estimate the ROA under a given control policy. Moreover, it is shown that the policy itself can also be updated with high probability guarantees on safety by iteratively learning the system dynamics [25]. The key idea of these studies is to construct a forward invariant safe region. If the system trajectory starts there, a valid control policy will exist to keep the future trajectory inside. Hence, the learning algorithm has the flexibility to explore and update its policy within the safe region, and safety is ensured as long as a corrective controller is applied when the system approaches the boundary of the safe region. However, for dynamical systems with high-dimensional state spaces, the solution to the partial differential equation in [22] might be hard to obtain, or the direct estimation of the ROA through sampling in [25] might become infeasible. Furthermore, estimating the unknown system dynamics or disturbances, those algorithms which rely on to provide safety guarantees, also poses difficulties. Acquiring a sufficient amount of data to get an accurate estimate as well as making feasible and suitable assumptions about the distribution of dynamics, e.g., which kernel function to be used in a Gaussian process, are challenging. In essence, it is not trivial to apply those methods for complex dynamical systems.

In most cases where a learning-based controller is desirable, it is likely that traditional controller design techniques might also be difficult to implement, mainly due to the complex dynamics. Hence, a safe learning approach should be capable of working on complex dynamical systems. Furthermore, finding the balance between the efficiency of exploration and the reliability of safety guarantees is critical. Although the learning performance can be improved by increasing the flexibility in the exploration process, more freedom might increase the risk of entering an unsafe state. A well-performed safe learning algorithm should maintain a high probability of safety while ensuring that the learning-based controller is not too restrictive. However, designing such an approach is challenging, especially when the system is high dimensional.

In this article, we propose a generalizable safe reinforcement learning framework that can be effectively used on complex dynamical systems. It allows autonomous systems to learn in the real world safer than standard learning methods, which in turn prevent damages to the system and the environment. With a proper definition of the safe region, the learning-based controller is able to execute the desired actions as long as the system is evolving inside the safe region. A predefined corrective controller is activated to keep the system inside the safe region when it reaches the safety boundary. By utilizing such a supervisory control strategy, the framework is compatible with arbitrary reinforcement learning algorithms, and is also applicable in various learning scenarios. For example, it can be used to increase the safety for transfer learning [10] or when a learning algorithm tries to improve the performance of a given controller [26]. In order to solve the challenging problem of dealing with complex dynamics, we utilize model order reduction techniques to construct a simplified system [27]. Such a simplified system facilitates an estimation for the safe region and enables a good initialization of the safe learning framework. To increase the flexibility of the learning-based controller, the safe region is modified through an online adaptation method. Due to the computational difficulty of calculating the exact safe region on complex dynamics [28], we have to relax the absolute safety guarantee by allowing a reasonable amount of failures. The proposed framework can learn effectively from those failures, and later avoid similar dangerous behaviors. During the adaptation, the execution results of the corrective controller provide feedback information to update the belief on maintaining the safety as well as the estimation of the safe region.

As the main contribution of this work, we provide a possible solution to the challenging problem of applying learning algorithms on complex dynamical systems in a safer and less restrictive manner, which enables application of state-of-the-art learning algorithms in real-world scenarios. Furthermore, we analyze the influence of the supervisory control strategy on the learning process through comprehensive simulations, which provide practical insights on how to achieve good balance between safety and learning performance.

The remainder of this article is organized as follows. A safe reinforcement learning framework that is directly implementable on low-dimensional dynamical systems is given in Section II. Thereafter, by using a model order reduction technique, the

practical realization of the framework on complex dynamical systems is proposed in Section III. In Section IV, two examples are given to demonstrate the performance of the proposed safe learning framework; followed by the discussion of several aspects of the framework in Section V. Finally, Section VI concludes this article.

II. Safe Learning Based on the ROA

In this section, a safe reinforcement learning framework based on the ROA is explained. Safe reinforcement learning aims to learn a control policy while satisfying safety constraints during the learning process. From the control theoretical perspective, one major criterion related to the safety of a dynamical system is stability. In that regard, if a given equilibrium point is known to be a safe state and is also locally asymptotically stable under a given control policy, then the ROA of this equilibrium point forms a safe region of the system. A supervisory control scheme, which switches between the learning-based controller and a predefined corrective controller, is constructed based on the safe region, such that the system remains safe during the training procedure. Details about the framework are presented in the remaining part of this section.

A. System Model and ROA

A general 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. For a given control policy $u = K(x) : \mathcal{X} \to \mathcal{U}$, the closed-loop system dynamics is denoted as

$$\dot{x} = f_K(x) \tag{2}$$

where $f_K(x) = f(x) + g(x)K(x)$. A state x is said to be an equilibrium point if $f_K(x) = 0$. Through a state transform, any equilibrium point can be shifted to the origin, hence in this work, we only focus on the ROA of the origin. The following assumptions are made for the system.

Assumption 1: $f_K(x)$ is Lipschitz continuous and bounded in $\mathcal{X}$.

Assumption 2: The origin is a locally asymptotically stable equilibrium point under $K(x)$.

Given these assumptions, the ROA of the origin under a given control policy $K(x)$ is defined as

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

where $\Phi_K(t; x_0)$ denotes the system trajectory of (2), which starts at initial state x_0 when time $t = 0$.

If the origin is a known safe state, then by utilizing the ROA, we define the safe region in this work as follows.

Definition 1: A closed positive invariant subset of the ROA $\mathcal{R}_K$ containing the origin is defined as the safe region $\mathcal{D}$ of the closed-loop dynamical system (2).

B. Safe Learning Framework

Typical reinforcement learning algorithms learn a parameterized control policy $u = \pi(x) : \mathcal{X} \to \mathcal{U}$ by iteratively updating its parameters through maximizing a reward function, whereas safe learning algorithms additionally require that certain safety constraints have to be satisfied during the learning process.

To formulate a safe reinforcement learning framework, a predefined corrective controller $K(x)$ and its safe region $\mathcal{D}$ are introduced to the learning process. As long as the system state x is inside the safe region $\mathcal{D}$, we can apply the control policy $K(x)$ to drive the system back to a safe state. Since $\mathcal{D}$ is a closed positive invariant set and the trajectory is continuous, the only possibility that the system leaves $\mathcal{D}$ is by crossing the boundary $\partial\mathcal{D}$. Thus, it is sufficient to apply $K(x)$ on the boundary $\partial\mathcal{D}$ to keep the system safe while providing flexible control action executions in the interior of $\mathcal{D}$.

Accordingly, we formulate the safe learning framework by switching the actual control action between the learning-based policy $\pi(x)$ and the corrective controller $K(x)$ for each learning iteration as

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

where t^* is the first time point where the system state x is on the boundary of the safe region $\partial\mathcal{D}$. For each learning iteration, the system state x starts inside the safe region $\mathcal{D}$ for time $t = 0$. The learning algorithm has the flexibility to evolve the trajectory within $\mathcal{D}$ and update the control policy $\pi(x)$. Once the system state x is on the boundary of the safe region, i.e., $x \in \partial\mathcal{D}$, this learning iteration is terminated at time $t = t^*$. The corrective controller $K(x)$ is then applied for the remaining time of this learning iteration to drive the system back to a safe state, i.e., the origin in our case. After the safety recovery, the learning environment is reset and the next learning iteration starts again with time $t = 0$. In practice, the time point t^* might be missed because of discretization of the control. In such cases, the time point t^* has to be modified by considering the tolerable distance to the boundary of the safe region with respect to the actual control frequency.

Such a supervisory control strategy is applicable to arbitrary reinforcement learning algorithms. We refer to this switching controller as the *supervisor* to the standard reinforcement learning structure (see Fig. 1). It examines the current system state x at every time step before applying the control action, and decides whether the learning-based controller has to be replaced by the corrective controller in order to keep the system safe. The volume of the safe region $\mathcal{D}$ depends on the corrective controller $K(x)$. A good choice of controller $K(x)$ provides a large enough $\mathcal{D}$ such that the system maintains sufficient flexibility under the policy $\pi(x)$.

One essential part of the safe learning framework is to represent the safe region as accurately as possible; however, the exact calculation of the safe region and the ROA is usually not feasible [29]. In literature, a sum-of-squares (SOS) programming approach is widely used to get at least an inner approximation of the ROA [30]. Since this approximation is

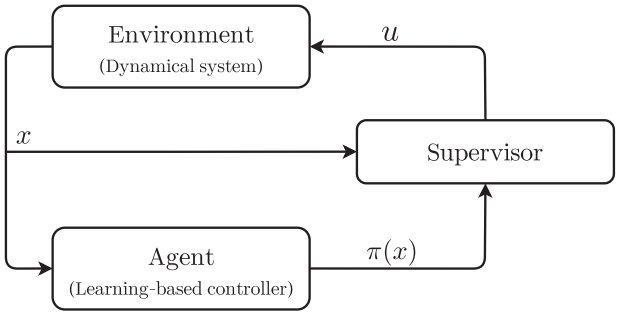


Fig. 1. Safe reinforcement learning framework with the supervisor that decides the actual control action applied on the system.

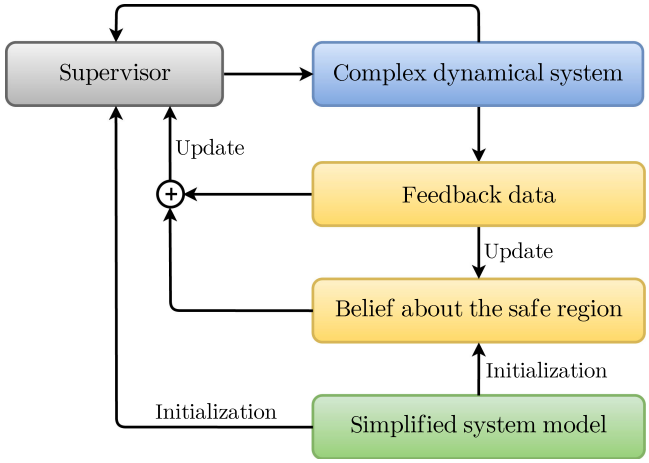


Fig. 2. Overview of the practical realization of the safe learning framework for complex dynamical systems. The supervisor and the belief about the safe region are initialized by using a simplified system model and are updated through feedback data.

represented as a closed positive invariant subset of the ROA, it serves as an initial estimate of the safe region. However, scalability is a major challenge in SOS programming [31], which hinders its application on complex dynamical systems. Although there exist methods to relax the computational limitation of SOS programming [31], it is still difficult to effectively update the high-dimensional approximation of the ROA obtained through these methods. Thus, in the next section, we propose a practically implementable safe learning framework for complex dynamical systems.

Remark 1: Using the ROA for defining the safe region provides computational efficiency; however, other concepts can also be utilized to define the safe region. For example, maximal control invariant set [32], invariance functions [33], or barrier functions [34] can be used to construct a safe region. The safe learning framework is generally compatible with these different concepts, as long as a closed and control invariant safe region can be defined.

III. Practical Realization on Complex Systems

In order to overcome the computational limitations, we propose a generalizable practical realization of the safe learning framework for complex dynamical systems in this section. An overview of the implementation is given in Fig. 2. We first utilize a model order reduction technique to extract a simplified system model from the complex dynamical system (denoted as the original system). Then, an approximation of the safe region of the simplified system is obtained through SOS programming. This approximation provides an initial belief about the safe region of the original system, and is used to initialize the supervisor of the safe learning framework. During the learning process, an online adaptation method is employed to update the supervisor as well as the belief about the true safe region. While the learning-based policy $\pi(x)$ is updated through reinforcement learning, a more reliable supervisor is learned simultaneously by using the feedback data from the execution of the corrective controller $K(x)$.

A. Initialization With Simplified System Model

One of the key properties of the safe learning framework is the identification of the safety of a state by the supervisor. Since no actual data are available prior to the learning process and finding the safe region of a complex dynamical system is computationally infeasible, the supervisor is initialized using the simplified system model.

1) Simplified System: In practical engineering problems, various model order reduction techniques are used when the direct operation over the original system is computationally infeasible [27], e.g., physically inspired approaches [35], balanced truncation [36], system immersion [37], or abstraction [38]. In this work, we perform the model order reduction by using physically inspired approaches. In general, if the original system is represented by (1), then by considering important physical features, the model order reduction finds a simplified system

$$\dot{x}_s = f_s(x_s) + g_s(x_s)u_s \tag{5}$$

where $x_s \in \mathbb{R}^{n_s}$, $n_s \ll n$ is the n_s-dimensional simplified system state and $u_s \in \mathbb{R}^{m_s}$ is the m_s-dimensional control input to the simplified system. The mapping of the model order reduction, which transfers the original system state x to the simplified system state x_s, is defined as $x_s = \Psi(x)$. For a given control policy $u_s = K_s(x_s)$, the safe region of the simplified system, denoted as $\mathcal{D}_s$, can be estimated through SOS programming [30].

For obtaining an approximation of the safe region of the original system $\mathcal{D}$, the following assumption is made.

Assumption 3: There exists a region around the original system's origin $\mathcal{V} = \{x \in \mathbb{R}^n \mid ||x|| \leq \varepsilon\} \subseteq \mathcal{D}$ with a constant $\varepsilon > 0$, such that $\forall x$, if $x \in \mathcal{V}$, then $\Psi(x) = x_s \in \mathcal{D}_s$.

Assumption 3 is a requirement on the quality of the combination of corrective controller $K(x)$ and simplified system. It says that if an original system state x is safe under the corrective controller $K(x)$, then the trajectory $\Phi_K(t; x)$ corresponds to a safe trajectory of the simplified system. Thus, it motivates the initialization of the safe region of the original system $\mathcal{D}$ by using the safe region of the simplified system $\mathcal{D}_s$. In general, if the original system state x corresponds to a safe simplified system state $x_s \in \mathcal{D}_s$, then a suitable corrective controller $K(x)$ is most

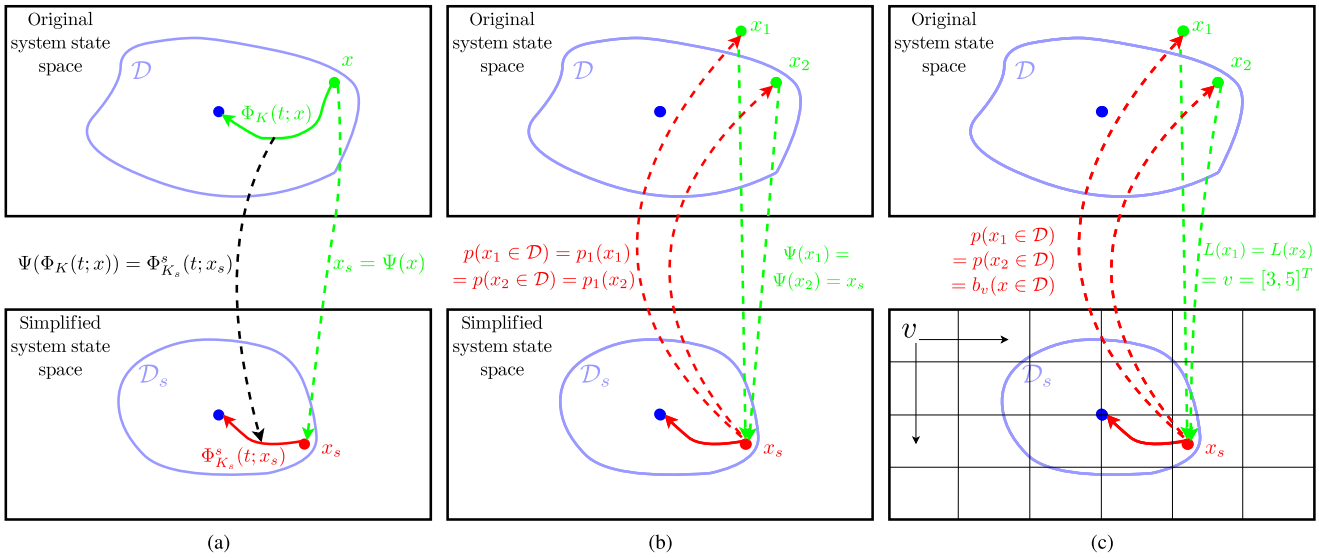


Fig. 3. Relationships between the safe regions of the original system and the simplified system. (a) For an original system state x and its corresponding simplified system state x_s, if $x_s \in \mathcal{D}_s$, then it is most likely that $x \in \mathcal{D}$ can be achieved by, e.g., solving $\Psi(\Phi_K(t;x)) = \Phi^s_{K_s}(t;x_s)$. (b) Different original system states x mapped to the same simplified system state x_s have the same probabilistic estimate of safety. (c) Probabilistic estimate of safety is represented by the corresponding belief mass.

probably also able to control the original system state back to a safe state. This can be achieved if there exists a corrective controller $K(x)$ that realizes ideal trajectory matching

$$\Psi(\Phi_K(t;x)) = \Phi^s_{K_s}(t;x_s) \tag{6}$$

where $\Phi^s_{K_s}(t;x_s)$ is the reference trajectory obtained from the simplified system that converges to the origin [see Fig. 3(a)]. See Remark 2 for an optimization-based approach. We will later discuss other choices for the corrective controller $K(x)$ that are found by physical insight or by considering technical limitations. Nevertheless, the simplified system provides a reasonable initialization of the safe learning framework.

In this work, we assume that a simplified system satisfying Assumption 3 is available, and a sufficiently accurate estimate of its safe region $\mathcal{D}_s$ is computable. $\mathcal{D}_s$ is then utilized as an initial belief of the safe region of the original system $\mathcal{D}$, which is modified later by an online adaptation method explained in the next section. A well-designed simplified system can provide an accurate belief that in turn reduces the amount of data required in the adaptation process.

Remark 2: To achieve (6), one possible choice for the corrective controller $K(x)$ is minimizing the discrepancy between the mapped trajectory $\Psi(\Phi_K(t;x))$ and the reference trajectory $\Phi^s_{K_s}(t;x_s)$. Hence, we have the following optimization problem

$$\min_u ||\nabla_x \Psi(x)(f(x) + g(x)u) - \dot{x}_s||, \text{ s.t. } u \in \mathcal{U} \tag{7}$$

where $\dot{x}_s = f_s(x_s) + g_s(x_s)K_s(x_s)$. However, since the original system has more degrees of freedom than the simplified system, in general, the set $\{u \mid \nabla_x \Psi(x)(f(x) + g(x)u) = \dot{x}_s\}$ is not a singleton. In such cases, additional constraints or costs are required to find the optimal control input u [33].

2) Probabilistic Estimate of Safety: Initially, the safety of an original system state x is estimated through the safety of its corresponding simplified system state $x_s = \Psi(x)$. However, there are three reasons for the inaccuracy when performing such an estimation, which are as follows.

1) Due to different control input spaces, the ideal trajectory matching represented by (6) may not be possible for all original system states x. Thus, the safety of an original system state x is not guaranteed by the safety of its corresponding simplified system state x_s (e.g., $\Psi(\Phi_K(t;x_1))$ in Fig. 4).
2) If an original system state x is mapped to a simplified system state x_s that is not in the safe region of the simplified system $\mathcal{D}_s$, the original system state x may still be a safe state (e.g., $\Psi(\Phi_K(t;x_2))$ in Fig. 4), since the control input of the original system u is usually more flexible than the control input of the simplified system u_s. However, in this case, prior knowledge about the original system is required to design the corrective controller $K(x)$ as trajectory matching is not suitable here.
3) Due to the order reduction, it is unavoidable that multiple original system states x are mapped to the same simplified system state x_s (e.g., $\Psi(x_1) = \Psi(x_3) = x_{s,1}$ in Fig. 4). Although these original system states x have the same estimate of safety from the simplified system, their actual safety properties are likely to be different, as the corrective controller provides different control signals for different original system states x.

Due to these reasons, we propose that for a given original system state x, the estimate of its safety obtained from the simplified system is represented in the following probabilistic

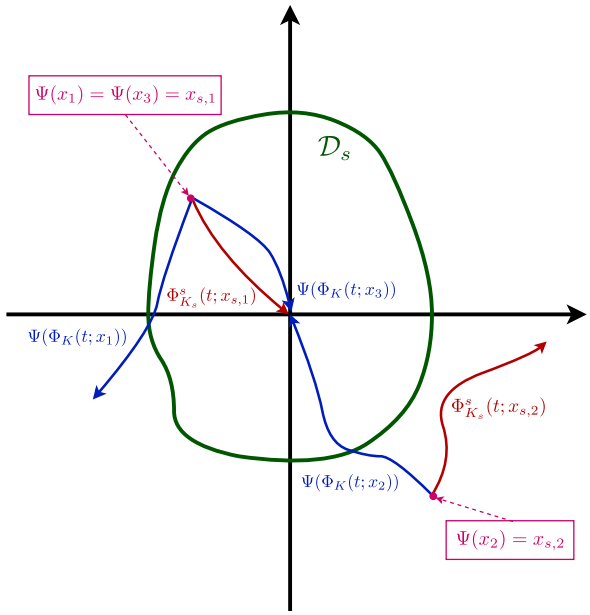


Fig. 4. Reasons for the inaccuracy of estimating the safety of an original system state through the simplified system. The original system state x does not always have the same safety property as its corresponding simplified system state x_s, and different original system states mapped to the same simplified system state may have different safety properties.

way:

$$p(x \in \mathcal{D}) = \underbrace{p(x \in \mathcal{D} | \Psi(x) \in \mathcal{D}_s)}_{p_1(x)} p(\Psi(x) \in \mathcal{D}_s) + \underbrace{p(x \in \mathcal{D} | \Psi(x) \notin \mathcal{D}_s)}_{p_2(x)} p(\Psi(x) \notin \mathcal{D}_s) \quad (8)$$

where different original system states x mapped to the same simplified system state $x_s = \Psi(x)$ have the same probabilities $p_1(x)$ and $p_2(x)$. As the safe region of the simplified system $\mathcal{D}_s$ is fixed, we have either $p(\Psi(x) \in \mathcal{D}_s) = 1$ or $p(\Psi(x) \notin \mathcal{D}_s) = 1$. Since $p(\Psi(x) \in \mathcal{D}_s) + p(\Psi(x) \notin \mathcal{D}_s) = 1$, (8) is simplified with the corresponding conditional probability as

$$p(x \in \mathcal{D}) = \begin{cases} p_1(x), & \text{if } \Psi(x) \in \mathcal{D}_s \\ p_2(x), & \text{if } \Psi(x) \notin \mathcal{D}_s \end{cases} \quad (9)$$

which represents the probabilistic estimate of safety of an original system state x [see Fig. 3(b)].

3) Supervisor Initialization: For the initialization of the supervisor, the only available information about the safe region of the original system $\mathcal{D}$ is the probabilistic estimate of safety derived from the simplified system. We therefore design a function approximator $F(x)$ as

$$F(x) = p(x \in \mathcal{D}) \sim [0, 1] \quad (10)$$

which gives the probability of a given original system state x being inside the safe region of the original system $\mathcal{D}$. Then, with a threshold α, the previous supervisor (4) is transformed into the following form:

$$u = \begin{cases} \pi(x), & \text{if } t < t' \\ K(x), & \text{else} \end{cases} \quad (11)$$

where t' denotes the first time point that the probability obtained from the function approximator is not larger than the threshold, i.e., $F(x) \leq \alpha$. Based on the probability of being in the safe region, the supervisor categorizes the original system states into safe and unsafe classes. Such a supervisor can be considered as a probabilistic binary classifier [39].

To initialize the function approximator $F(x)$ of the supervisor, we first sample k normally distributed points in the original system state space $\{x_1, x_2, \ldots, x_i, \ldots, x_k\}$, where k depends on the dimension of the state space. Then, according to (9), the probabilistic estimate of safety of each sample is calculated by using the safe region of the simplified system $\mathcal{D}_s$. We set the probability distributions to constants $p_1(x) = \overline{p}, p_2(x) = \underline{p}$, and obtain the initial estimate

$$p_{\text{init}}(x_i \in \mathcal{D}) = \begin{cases} \overline{p}, & \text{if } \Psi(x_i) \in \mathcal{D}_s \\ \underline{p}, & \text{if } \Psi(x_i) \notin \mathcal{D}_s \end{cases} \quad (12)$$

where $0 < \underline{p} < \alpha < \overline{p} < 1$. The values of $\overline{p}$ and $\underline{p}$ are chosen to be the same as in the prior belief map, which is explained in Section III-B.3. An example is given in the following.

Example 1: Assume the original system state is $x = [x_a, x_b, x_c, x_d]^T \in \mathcal{X} \subseteq \mathbb{R}^4$, we define the simplified system state $x_s \in \mathbb{R}^2$ and the safe region of the simplified system $\mathcal{D}_s$ as

$$x_s = \Psi(x) = [x_e, x_f]^T = [x_a + x_b, x_c + x_d]^T \quad (13)$$

$$\mathcal{D}_s = \{x_s | x_e^2 + x_f^2 \leq 9\}. \quad (14)$$

Then, for one sampled state $x_1 = [0.1, 0.2, 0.3, 0.4]^T$, we have $p_{\text{init}}(x_1 \in \mathcal{D}) = \overline{p}$ since $x_{s,1} = \Psi(x_1) = [0.3, 0.7]^T \in \mathcal{D}_s$.

The function approximator $F(x)$ is then initialized by training with all sampled states and their corresponding estimates obtained from (12). In general, this initialization is expected to result in a restricted supervisor. To increase the performance of the safe learning framework, we update the supervisor with an online adaptation method that is described in the next section.

Remark 3: Depending on the sample size k, we use either a Gaussian process regression (GPR) model or a neural network (NN) for the function approximator $F(x)$ in this work. Although training a GPR model is more transparent compared to NN, it has computational difficulties when dealing with a large dataset. Therefore, to enable efficient training, we use an NN when the original system state space has dimensions higher than 8, where usually more than a few thousand data points are required for a reliable approximation. The function approximator $F(x)$ can also be represented by other models, as long as reasonable predictions can be made and efficient training is feasible.

B. Online Adaptation of the Supervisor

During the learning process, a learning iteration is terminated when the supervisor realizes that the trajectory is on the boundary of the estimated safe region of the original system $\mathcal{D}$. The predefined corrective controller $K(x)$ is then activated to

recover safety by controlling the original system state x back to the origin. The success of this recovery informs us about the ground-truth value of safety and is defined as feedback data.

Definition 2: The feedback data of a learning iteration that requires safety recovery contain two elements $(x, p(x \in \mathcal{D}) = \{1, 0\})$. x is the original system state where the corrective controller $K(x)$ is activated. $p(x \in \mathcal{D}) = \{1, 0\}$ represents whether this state can be controlled back to the origin under the given corrective controller $K(x)$. We call it positive feedback data if $p(x \in \mathcal{D}) = 1$, and negative feedback data if $p(x \in \mathcal{D}) = 0$. The set of all feedback data is denoted as X_{real}.

Ideally, the supervisor is updated by only using feedback data so that if enough trials are available, the supervisor will provide accurate predictions. However, for a system with high-dimensional state space, it is not feasible to acquire such an amount of feedback data through real trials. The proposed online adaptation method tackles this problem. The central idea is to update the supervisor by a hybrid data source, comprising the feedback data and the probabilistic estimate of safety obtained from the simplified system. For getting accurate estimates, the belief about the safe region of the original system $\mathcal{D}$ is updated during the learning process such that a reliable supervisor is obtained through the online adaptation method.

1) Belief Function Theory: The probabilistic estimate of safety obtained from (9) comes from the simplified system. However, as the exact mathematical relationship between the safe region of the original system $\mathcal{D}$ and the safe region of the simplified system $\mathcal{D}_s$ is unclear, such an estimate is in fact a subjective probability [40]. Moreover, for obtaining accurate estimates, the probability distributions $p_1(x)$ and $p_2(x)$ need to be updated using feedback data. Due to the insufficiency of feedback data, there is an internal uncertainty affecting the accuracy of the update. For example, when tossing a coin, the probabilities of "heads" and "tails" are equal, i.e., $p(\text{heads}) = p(\text{tails}) = 0.5$. But with only one toss, the probability we obtain from the observation is either $p(\text{heads}) = 1$ or $p(\text{tails}) = 1$. The same problem occurs when computing probability from insufficient feedback data, which makes it also a subjective probability.

To deal with subjective probability, we integrate the belief function theory [41] into our safe learning framework. Belief function theory provides a general approach for modeling epistemic uncertainty by using belief mass and basic belief assignment (BBA). While belief mass represents the probability of the occurrence of an event, BBA denotes the assignment of belief masses to all possible events. The subjective uncertainty is included as a belief mass on the entire event domain, i.e., the probability that one arbitrary event happens [42]. Therefore, we reformulate the probabilistic estimate of safety into belief mass and design the online adaptation method accordingly, this allows to adjust the belief about the safe region by accounting for the subjective uncertainty.

2) Belief Map: In order to efficiently employ the belief function theory, the simplified system state space is discretized into grid cells. Each grid cell is indexed by an index vector $v \in \mathbb{Z}_+^{n_s}$ that indicates its location in the simplified system state space. We then define a labeling function $L(x)$.

Definition 3: For an original system state x, the labeling function $L(x)$ returns the index vector v of the grid cell that the corresponding simplified system state $x_s = \Psi(x)$ lies in.

We assume that all original system states x, which have the same index vector v from the labeling function $L(x)$, i.e., they map to the same grid cell in the simplified system state space, have the same probabilistic estimate of safety. For taking the subjective probability into consideration, we design a belief map that transforms (9) to belief masses.

Definition 4: A belief map $B(v)$ that assigns a BBA to each index vector v is defined as

$$B(v) = (b_v(x \in \mathcal{D}), b_v(x \notin \mathcal{D}), \sigma_v) \tag{15}$$

where $b_v(x \in \mathcal{D})$ and $b_v(x \notin \mathcal{D})$ are belief masses, σ_v is the subjective uncertainty. For each BBA, it holds

$$b_v(x \in \mathcal{D}) + b_v(x \notin \mathcal{D}) + \sigma_v = 1 \tag{16}$$

and $b_v(x \in \mathcal{D}), b_v(x \notin \mathcal{D}), \sigma_v$ lie within the interval [0,1].

The belief masses $b_v(x \in \mathcal{D})$ and $b_v(x \notin \mathcal{D})$ within each BBA represent the probabilities of the occurrence of two complementary events $x \in \mathcal{D}$ and $x \notin \mathcal{D}$, i.e., given the fact that the original system state x has the index vector v from the labeling function $L(x)$, whether x is in the safe region of the original system $\mathcal{D}$ or not. The subjective uncertainty σ_v reflects the confidence level of making such a probabilistic estimate. $\sigma_v = 0$ means we believe that the estimate is absolutely correct. To simplify the notations, we denote $b_v(x \in \mathcal{D})$ as $b_{v,s}$ and $b_v(x \notin \mathcal{D})$ as $b_{v,u}$ for belief masses of the safe and unsafe events.

The belief map $B(v)$ utilizes the index vector v to make the probabilistic estimate of safety for different original system states x. The probability distributions $p_1(x)$ and $p_2(x)$ in (9) are discretized and replaced accordingly with the belief mass $b_{v,s}$ for each index vector v. For an original system state x, the probabilistic estimate of safety obtained from the belief map $B(v)$ is given as

$$p(x \in \mathcal{D}) = b_v(x \in \mathcal{D}) = b_{v,s} \tag{17}$$

where the corresponding index vector $v = L(x)$ is determined by the labeling function $L(x)$ [see Fig. 3(c)]. An example of the belief map $B(v)$ is given as follows.

Example 2: Continued from Example 1, we assume that all original system states $x \in \mathcal{X}$ have their simplified system states x_s in $\mathcal{X}_s = \{x_s \mid -4 \leq x_e \leq 4, -4 \leq x_f \leq 4\}$. After the discretization with step size 1 for both x_e and x_f in the simplified system state space, an index vector $v = [v_f, v_e]^T, v_f, v_e \in \{1, 2, \ldots, 8\}$ is assigned to each grid cell (see Fig. 5). For original system states $x_1 = [0.1, 0.2, 0.3, 0.4]^T$, $x_2 = [0.2, 0.3, 0.4, 0.5]^T$, and $x_3 = [0, 1.5, 1, 1.5]^T$, by locating the corresponding simplified system states $x_{s,1} = [0.3, 0.7]^T$, $x_{s,2} = [0.5, 0.9]^T$, and $x_{s,3} = [1.5, 2.5]^T$, we thus have $L(x_1) = L(x_2) = [4, 5]^T$ and $L(x_3) = [2, 6]^T$. The probabilistic estimates of safety are therefore given as $p(x_1 \in \mathcal{D}) = p(x_2 \in \mathcal{D}) = b_{[4,5]^T,s}$ and $p(x_3 \in \mathcal{D}) = b_{[2,6]^T,s}$.

3) Prior and Feedback Belief Map: As mentioned earlier, the subjective uncertainties are caused by the unknown reliability of the simplified system and the insufficient amount of feedback

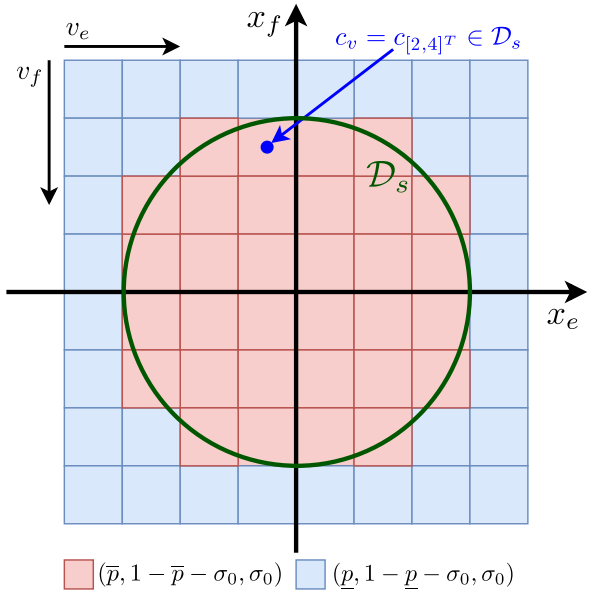


Fig. 5. Prior belief map $B^{\mathrm{S}}(v)$ of the examples. The BBA for each index vector v is determined by the center point c_v of the corresponding grid cell.

data. In the proposed online adaptation method, we consider these two types of subjective uncertainties with two different belief maps.

For the subjective uncertainty in the simplified system, we construct a prior belief map $B^{\mathrm{S}}(v)$ with the constants $\overline{p}$ and $\underline{p}$ from (12) as

$$B^{\mathrm{S}}(v) = \left(b^{\mathrm{S}}_{v,s}, b^{\mathrm{S}}_{v,u}, \sigma^{\mathrm{S}}_v\right) = \begin{cases} (\overline{p}, 1-\overline{p}-\sigma_0, \sigma_0), & \text{if } c_v \in \mathcal{D}_s \\ (\underline{p}, 1-\underline{p}-\sigma_0, \sigma_0), & \text{if } c_v \notin \mathcal{D}_s \end{cases} \tag{18}$$

where for each index vector v, the BBA is determined by identifying if the center point c_v of the corresponding grid cell lies inside the safe region of the simplified system $\mathcal{D}_s$ (e.g., $v = [2,4]^T$ in Fig. 5). The subjective uncertainty σ^{S}_v reflects the reliability of the simplified system and is set to a constant σ_0 for all index vectors v. The superscript S stands for the simplified system as the belief source.

The prior belief map $B^{\mathrm{S}}(v)$ represents the initial belief about the safe region of the original system $\mathcal{D}$ obtained from the simplified system. As the simplified system is fixed during the learning process, we keep the prior belief map $B^{\mathrm{S}}(v)$ unchanged.

The subjective uncertainty in feedback data decreases with more observations. Accordingly, we first initialize two counters $C_p(v)$ and $C_n(v)$ to zero for every index vector v. For each index vector v, $C_p(v)$ counts the number of positive feedback data (similarly, $C_n(v)$ for negative feedback data) whose original system state x maps to the grid cell indexed by v, i.e., $L(x) = v$. Every time the corrective controller $K(x)$ has to be applied for an original system state x, the counters $C_p(v)$ and $C_n(v)$ are updated based on the corresponding feedback data as

$$\begin{cases} C^+_p(v) = C_p(v) + 1, & \text{if } p(x \in \mathcal{D}) = 1 \\ C^+_n(v) = C_n(v) + 1, & \text{if } p(x \in \mathcal{D}) = 0 \end{cases} \tag{19}$$

with the index vector $v = L(x)$ obtained from the labeling function $L(x)$.

By utilizing the counters $C_p(v)$ and $C_n(v)$, we therefore construct another feedback belief map $B^{\mathrm{F}}(v)$, where the superscript F indicates that the belief is based on the feedback data. For each index vector v, the BBA obtained from the feedback belief map $B^{\mathrm{F}}(v)$ is defined as

$$B^{\mathrm{F}}(v) = (b^{\mathrm{F}}_{v,s}, b^{\mathrm{F}}_{v,u}, \sigma^{\mathrm{F}}_v) \tag{20}$$

where if at least one feedback data is available for the given index vector v, i.e., $C_p(v) + C_n(v) > 0$, we let

$$b^{\mathrm{F}}_{v,s} = \frac{C_p(v)}{C_p(v) + C_n(v)}(1 - \sigma^{\mathrm{F}}_v) \tag{21}$$

$$b^{\mathrm{F}}_{v,u} = \frac{C_n(v)}{C_p(v) + C_n(v)}(1 - \sigma^{\mathrm{F}}_v) \tag{22}$$

$$\sigma^{\mathrm{F}}_v = c_1 \cdot e^{-c_2 \cdot (C_p(v) + C_n(v) - 1)} \tag{23}$$

otherwise we set $B^{\mathrm{F}}(v)$ to an empty BBA. The subjective uncertainty σ^{F}_v depends on the amount of feedback data. If sufficient data are obtained, the subjective uncertainty σ^{F}_v approaches 0, and the belief mass $b^{\mathrm{F}}_{v,s}$ becomes the actual probability. Coefficients c_1 and c_2 define the initial value and the decay rate of the uncertainty, respectively.

During the learning, the counters $C_p(v)$ and $C_n(v)$ are iteratively modified when new feedback data are obtained. Whenever an update of the supervisor is needed, the feedback belief map $B^{\mathrm{F}}(v)$ is calculated from the up-to-date counters $C_p(v)$ and $C_n(v)$.

While the prior belief map $B^{\mathrm{S}}(v)$ stands for the model-based belief, the feedback belief map $B^{\mathrm{F}}(v)$ represents the data-driven belief. For getting a more accurate belief about the safe region of the original system $\mathcal{D}$, in the following, we introduce a belief fusion operation to combine these two belief maps.

Remark 4: The subjective uncertainty σ^{F}_v in (23) can be represented in different forms, as long as it lies within the interval [0,1] and satisfies that the uncertainty decreases with more feedback data.

4) Weighted Belief Fusion: Combining the prior belief map $B^{\mathrm{S}}(v)$ and the feedback belief map $B^{\mathrm{F}}(v)$ is referred to as weighted belief fusion [43], which results in a combined belief map $B^{\mathrm{C}}(v)$. For each index vector v, the BBA of the combined belief map $B^{\mathrm{C}}(v)$ is defined as

$$B^{\mathrm{C}}(v) = (b^{\mathrm{C}}_{v,s}, b^{\mathrm{C}}_{v,u}, \sigma^{\mathrm{C}}_v) = \begin{cases} B^{\mathrm{S}}(v), & \text{if } B^{\mathrm{F}}(v) \text{ is empty} \\ B^{\mathrm{S}}(v) \oplus B^{\mathrm{F}}(v), & \text{if } B^{\mathrm{F}}(v) \text{ is not empty} \end{cases} \tag{24}$$

where the operator $\oplus$ is given as

$$b^{\mathrm{C}}_{v,s} = \frac{b^{\mathrm{S}}_{v,s}(1-\sigma^{\mathrm{S}}_v)\sigma^{\mathrm{F}}_v + b^{\mathrm{F}}_{v,s}(1-\sigma^{\mathrm{F}}_v)\sigma^{\mathrm{S}}_v}{\sigma^{\mathrm{S}}_v + \sigma^{\mathrm{F}}_v - 2\sigma^{\mathrm{S}}_v\sigma^{\mathrm{F}}_v} \tag{25}$$

$$b^{\mathrm{C}}_{v,u} = \frac{b^{\mathrm{S}}_{v,u}(1-\sigma^{\mathrm{S}}_v)\sigma^{\mathrm{F}}_v + b^{\mathrm{F}}_{v,u}(1-\sigma^{\mathrm{F}}_v)\sigma^{\mathrm{S}}_v}{\sigma^{\mathrm{S}}_v + \sigma^{\mathrm{F}}_v - 2\sigma^{\mathrm{S}}_v\sigma^{\mathrm{F}}_v} \tag{26}$$

$$\sigma^{\mathrm{C}}_v = \frac{(2-\sigma^{\mathrm{S}}_v-\sigma^{\mathrm{F}}_v)\sigma^{\mathrm{S}}_v\sigma^{\mathrm{F}}_v}{\sigma^{\mathrm{S}}_v + \sigma^{\mathrm{F}}_v - 2\sigma^{\mathrm{S}}_v\sigma^{\mathrm{F}}_v}. \tag{27}$$

The combined belief map $B^{\mathrm{C}}(v)$ satisfies the following properties.

Proposition 1: If a sufficiently large set of feedback data is provided, the combined belief map $B^{\mathrm{C}}(v)$ converges to the actual probabilities and the prior belief map $B^{\mathrm{S}}(v)$ has no effect in making estimates.

Proof: The following holds for the weighted belief fusion:

$$\begin{aligned}
&\lim_{C_p(v)+C_n(v)\to\infty} b^{\mathrm{C}}_{v,s} = b^{\mathrm{F}}_{v,s} \\
&\lim_{C_p(v)+C_n(v)\to\infty} b^{\mathrm{C}}_{v,u} = b^{\mathrm{F}}_{v,u} \\
&\lim_{C_p(v)+C_n(v)\to\infty} \sigma^{\mathrm{C}}_v = \sigma^{\mathrm{F}}_v = 0
\end{aligned} \tag{28}$$

which directly leads to Proposition 1. ■

An example is given as follows.

Example 3: Continued from previous examples, we now observe two feedback data $(x_1, p(x_1) = 1)$, $(x_2, p(x_2) = 0)$ with $x_1 = [0.1, 0.2, 0.3, 0.4]^T$ and $x_2 = [0.2, 0.3, 0.4, 0.5]^T$. Then, for the index vector $v = [4, 5]^T$, we have $C_p([4, 5]^T) = 1$, $C_n([4, 5]^T) = 1$, and the corresponding BBA obtained from the feedback belief map $B^{\mathrm{F}}(v)$ is

$$B^{\mathrm{F}}([4, 5]^T) = (0.47, 0.47, 0.06) \tag{29}$$

if $c_1 = 0.1$ and $c_2 = 0.5$. Assume that the prior belief map $B^{\mathrm{S}}(v)$ is constructed with $\overline{p} = 0.8$, $\underline{p} = 0.1$, and $\sigma_0 = 0.1$, we obtain

$$B^{\mathrm{S}}([4, 5]^T) = (0.8, 0.1, 0.1). \tag{30}$$

Therefore, the BBA of the combined belief map $B^{\mathrm{C}}(v)$ for $v = [4, 5]^T$ is calculated as

$$B^{\mathrm{C}}([4, 5]^T) = (0.59, 0.34, 0.07). \tag{31}$$

The combined belief map $B^{\mathrm{C}}(v)$ is then used to make the probabilistic estimate of safety for other original system states, e.g., for $x_4 = [0.4, 0.3, 0.2, 0.1]^T$, we have $p(x_4 \in \mathcal{D}) = b^{\mathrm{C}}_{[4,5]^T,s} = 0.59$.

In essence, the prior belief map $B^{\mathrm{S}}(v)$ serves as the initial belief that facilitates the safe learning framework in the earlier phases. During the learning process, such an initial belief is improved through weighted belief fusion. With more available feedback data, the combined belief map $B^{\mathrm{C}}(v)$ gets closer to the actual probabilities, and therefore, a more accurate probabilistic estimate of safety is obtained. The update of the supervisor is thus performed by using the combined belief map $B^{\mathrm{C}}(v)$.

Parameters of the framework are chosen by considering the following aspects.

1) The threshold α decides on the aggressiveness of actions and is selected based on the actual task and the severity of failure.
2) The initialization of the prior belief map $B^{\mathrm{S}}(v)$ depends on the representation power of the simplified model. For an informative simplified model, a high probability is assigned to $\overline{p}$ along with a low uncertainty σ_0. $\underline{p}$ can be chosen as $1 - \overline{p} - \sigma_0$ such that it represents the complementary probability of an initial unsafe estimate. In general, to ensure an adequate initial action space, $\overline{p}$ needs to be larger than α.
3) The parameters of the feedback belief map $B^{\mathrm{F}}(v)$ are selected by considering the number of positive feedback data needed to convert an initial unsafe estimate to a safe estimate. For example, if three positive feedback data are required, then c_1 and c_2 satisfy that $b^{\mathrm{C}}_{v,s} > \alpha$ is obtained from (25) with $C_p(v) = 3$ and $C_n(v) = 0$.

5) Supervisor Update: Although the set of feedback data X_{real} contains the ground-truth information about the safe region of the original system $\mathcal{D}$, its limited size makes it inefficient for the update of the supervisor. We solve this problem by generating another set of auxiliary data using the combined belief map $B^{\mathrm{C}}(v)$. Considering the computational efficiency, we perform one update of the supervisor whenever k_u feedback data are obtained, where the value of k_u depends on the task.

Similar to the initialization of the supervisor in Section III-A.3, we first sample k_e normally distributed original system states $\{x_1, x_2, \ldots, x_i, \ldots, x_{k_e}\}$. To prevent repetition with the set of feedback data X_{real}, the samples have to be not in the neighborhood of any known states with a distance threshold d, i.e., $\forall x \in X_{\mathrm{real}}$, it holds $||x - x_i|| > d$ with $i = 1 \ldots k_e$. Then, according to (17), the probabilistic estimate of safety is assigned to each sample by using the combined belief map $B^{\mathrm{C}}(v)$ as

$$p(x_i \in \mathcal{D}) = b^{\mathrm{C}}_{v_i,s} \tag{32}$$

where the index vector $v_i = L(x_i)$ is computed by the labeling function $L(x)$. This set of estimates is denoted as X_{est} and is used to support the update of the supervisor. While the set of feedback data X_{real} represents the absolute safety of already known states, X_{est} predicts the safety of unknown states. In each supervisor update iteration, a new set of estimates X_{est} is created with the up-to-date combined belief map $B^{\mathrm{C}}(v)$. When the accuracy of the combined belief map $B^{\mathrm{C}}(v)$ is improved with more feedback data, more reliable estimates are obtained.

The update of the supervisor is performed by training a new function approximator $F(x)$ from the combined dataset that includes the set of feedback data X_{real} and the set of estimates X_{est}. In the learning process, we incrementally extend the set X_{real} after new feedback data are obtained. The size k_e of the set of estimates X_{est} is set to a fixed value since X_{est} only works as supplementary data.

Although the combined belief map $B^{\mathrm{C}}(v)$ converges to the actual probabilities with more data points, acquiring a sufficient amount of data is usually not feasible in practice. Therefore, it is unavoidable that some estimates are incorrect, especially in the early stage of learning. Estimates with a high error rate will not

only result in a poorly performing supervisor, but also indicate that the prior belief map $B^{\mathrm{S}}(v)$ constructed from the simplified system is not reliable. In such a case, a considerable amount of feedback data is required until the inaccuracy of the prior belief map $B^{\mathrm{S}}(v)$ is offset through the weighted belief fusion. As a result, our original intention of using the simplified system to provide initial belief about the safe region becomes insignificant. To prevent this, we additionally introduce a validation process into the safe learning framework.

6) Validation: The validation aims to identify the accuracy of the combined belief map $B^{\mathrm{C}}(v)$. During the learning process, we perform the validation prior to the update of the supervisor. Since the supervisor works as a binary classifier, we use the confusion matrix obtained from the classification for the validation.

For each original system state x in feedback data, the probabilistic estimate of safety is obtained by using the combined belief map $B^{\mathrm{C}}(v)$ with (32). Then, according to the estimate, a predicted class label is assigned to each feedback data by categorizing it into safe and unsafe classes with respect to the same threshold α in (11). Comparing with the true class label of feedback data, i.e., $p(x \in \mathcal{D}) = \{1, 0\}$, we acquire the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) data points, respectively. The reliability of the combined belief map $B^{\mathrm{C}}(v)$ is therefore represented by the accuracy $\mathrm{ACC} = (\mathrm{TP} + \mathrm{TN})/(\mathrm{TP} + \mathrm{TN} + \mathrm{TP} + \mathrm{TN})$ and the FP ratio $\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{TN})$.

Since the corrective controller is applied when the supervisor considers the current state to be on the boundary of the estimated safe region, the set of feedback data X_{real} in fact only contains information about the boundary. In that sense, the validation process also examines whether the combined belief map $B^{\mathrm{C}}(v)$ can represent the real decision boundary accurately. For a successful implementation of the proposed framework, the simplified system should fulfill the following assumption.

Assumption 4: The safe and unsafe regions of the original system are separable in the simplified system state space, i.e., the decision boundary exists.

The accuracy (ACC) and the FP ratio (FPR) reflect how well Assumption 4 is satisfied. To ensure the performance of the supervisor, the combined belief map $B^{\mathrm{C}}(v)$ should possess a high ACC and a small FPR. Otherwise, the safe learning process is suggested to be terminated and restarted with another simplified system model.

One possible way to find a better simplified system model is examining the original system states that cannot be distinguished by the current simplified system model, i.e., they have the same index vector v but show different safety properties. The state variables (features) that differ significantly between these original system states could be incorporated into the new simplified system model. However, this increases the complexity of the simplified system.

C. Safe Learning Algorithm

The practical realization of the proposed safe learning framework is given in Algorithm 1. During the learning process, a learning iteration is terminated if the supervisor believes that the current state is on the boundary of the estimated safe region. The corrective controller $K(x)$ is then applied to attempt to drive the system back to the origin. The learning-based policy $\pi(x)$ is updated based on a predefined reward function, whereas the supervisor is updated according to the feedback data. Once new feedback data are obtained, the set of feedback data X_{real} is extended, and the counters $C_p(v)$ and $C_n(v)$ are updated. To prevent repetition, this modification only happens if all original system states stored in X_{real} have distances to the new feedback data's state x larger than a threshold d, or the new feedback data represent a failure. Whenever k_u feedback data are obtained, the prior belief map $B^{\mathrm{S}}(v)$ and the feedback belief map $B^{\mathrm{F}}(v)$ are

Algorithm 1: Practical Realization of the Proposed Safe Learning Framework.

Input: Safe region of the simplified system $\mathcal{D}_s$, probability threshold α, distance threshold d, initialization sampling size k, estimation sampling size k_e, number of learning iterations k_i, supervisor update interval k_u ;

```
Initialize F(x) with D_s and k ;
Initialize B^S(v) with D_s ;
Initialize C_p(v) and C_n(v) ;
Set X_real as an empty set;
i = 0, j = 0 ;
while i < k_i do
    while current learning iteration is not terminated do
        if F(x) > α then
            u = π(x) ;
        else
            Iteration is terminated ;
        end
    end
    Update π(x) ;
    Apply K(x) ;
    if ∀x_i ∈ X_real, ||x − x_i|| > d or p(x ∈ D) = 0 then
        Append X_real with (x, p(x ∈ D) = {0, 1}) ;
        Update C_p(v) or C_n(v) ;
        j = j + 1 ;
    end
    if j = k_u then
        Calculate B^F(v) from C_p(v) and C_n(v) ;
        Calculate B^C(v) ;
        Validation ;
        if Valid then
            Create X_est with B^C(v) and k_e, d ;
            Update F(x) with X_real and X_est ;
            j = 0 ;
        else
            Safe learning process is suggested to be
             terminated ;
        end
    end
    i = i + 1 ;
end
```

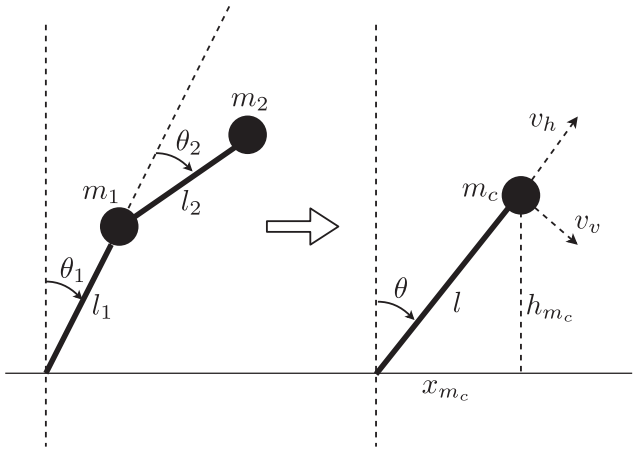


Fig. 6. Model order reduction for a two-link inverted pendulum. By using the CoM, a one-link inverted pendulum approximates the two-link inverted pendulum dynamics.

employed to form the combined belief map $B^{\mathrm{C}}(v)$. The set of estimates X_{est} is then generated by using the combined belief map $B^{\mathrm{C}}(v)$. A new function approximator $F(x)$ is therefore trained with the combined dataset, and formulates an updated supervisor for the following learning iteration.

With the proposed algorithm, we are able to realize a feasible implementation of the safe learning framework on complex dynamical systems. The supervisor proposed in this work provides predictions about the safety in an effective way. The results of this practical yet accurate framework are demonstrated in the next section.

IV. Simulations and Experiments

In this section, two examples are presented. First, a simple two-link inverted pendulum example illustrates how the supervisor and the belief map are updated based on the feedback data. Second, in a quadcopter control task, the performance of the proposed safe learning framework for complex dynamical systems is demonstrated.

A. Two-Link Inverted Pendulum

We first demonstrate the online adaptation of the supervisor for a two-link inverted pendulum given in Fig. 6. It is assumed that $l_1 = l_2, m_1 = m_2$, and the links have no masses. The system state combines the two joint angles and the angular velocities, i.e., $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$, and the control input u is the torques applied on the links. The origin, which is the upright equilibrium point, is considered as the safe state. A simple PID controller is implemented as the corrective controller $K(x)$. By limiting the input torques, the inverted pendulum cannot be driven back to the origin by the given corrective controller $K(x)$ once it exceeds a certain angle. The purpose of the online adaptation is therefore to find a supervisor that can accurately represent the safe region of the original system, i.e., the two-link inverted pendulum.

By using a physically inspired model order reduction, a one-link inverted pendulum is constructed based on the center of mass (CoM) m_c and is considered as the simplified system (see Fig. 6). The simplified system state is $x_s = [\theta, \dot{\theta}]^T$ and the corresponding mapping $x_s = \Psi(x)$ is

$$\theta = \arctan \frac{x_{m_c}}{h_{m_c}}, \quad \dot{\theta} = \frac{v_v}{l} \tag{33}$$

where the link length l is assumed to be fixed to $l = l_1 + \frac{l_2}{2}$. The joint angle θ is determined from the horizontal and vertical positions of the CoM, x_{m_c} and h_{m_c}, and the angular velocity $\dot{\theta}$ is obtained by the velocity of the CoM. Note that the velocity of the CoM has two components, one is along the link v_h and the other is perpendicular to the link v_v. As the link length l is considered to be fixed, v_h is infeasible for the one-link inverted pendulum and thus has to be neglected in the mapping. The omission of v_h represents the expected drawback of losing information when the system order is reduced in terms of the dimensionality of the state space.

Due to the simplicity of this example, the safe region of the simplified system $\mathcal{D}_s$ obtained from SOS programming provides an accurate estimate about the safe region of the original system $\mathcal{D}$. To demonstrate the online adaptation clearly, instead of using SOS programming, we select the safe region of the simplified system as $\mathcal{D}_s = \{x_s \mid \theta^2 + \dot{\theta}^2 \leq 0.6^2\}$, where the units of θ and $\dot{\theta}$ are radian and radian per second, respectively. The discrepancies caused by this selection are compensated later by the feedback data. Moreover, no learning algorithm is implemented in this example. Since if a learning-based controller is introduced, then under the given input constraints, most of the grid cells in the simplified state space cannot be visited through a natural movement of the system. For example, there exists no admissible control signal to drive the system to states that have a large positive angle θ and a large negative angular velocity $\dot{\theta}$ at the same time. Therefore, to fully illustrate the update process of the combined belief map $B^{\mathrm{C}}(v)$, the corrective controller $K(x)$ is activated at randomly selected original system states x such that more states can be visited.

The parameters of the safe learning framework used in this example are given in Appendix B. For the two-link inverted pendulum, we set $l_1 = l_2 = 1$ m and $m_1 = m_2 = 1$ kg. The input torque limit is selected as 12 N for both links. The simplified system state space $x_s = [\theta, \dot{\theta}]^T$ is assumed to be within the range of $\theta \in [-1.6\,\mathrm{rad}, 1.6\,\mathrm{rad}]$, $\dot{\theta} \in [-2\,\mathrm{rad/s}, 2\,\mathrm{rad/s}]$, and is discretized with 0.1 rad for θ and 0.1 rad/s for $\dot{\theta}$.

The corresponding results are shown in Fig. 7. The first row of the figure gives the probability $p(x \in \mathcal{D})$ from the function approximator $F(x)$ for the slice $\dot{\theta}_1 = \dot{\theta}_2 = 0$ in different supervisor update iterations, whereas the second row shows the slice $\theta_2 = \dot{\theta}_2 = 0$. The ground-truth values of safety $p(x \in \mathcal{D}) = \{0, 1\}$ are presented for comparison. The last row illustrates the belief mass $b^{\mathrm{C}}_{v,s}$ from the combined belief map $B^{\mathrm{C}}(v)$ for different index vectors v. In the first supervisor update iteration, i.e., the initialization, the combined belief map $B^{\mathrm{C}}(v)$ is equal to the prior belief map $B^{\mathrm{S}}(v)$. Due to the inaccuracy of the prior belief map $B^{\mathrm{S}}(v)$, the prediction differs from the ground-truth value [see Fig. 7(a), (e), and (i)]. However, after ten updates of the supervisor, the accuracy of the prediction increases quickly. The combined belief map $B^{\mathrm{C}}(v)$ also changes

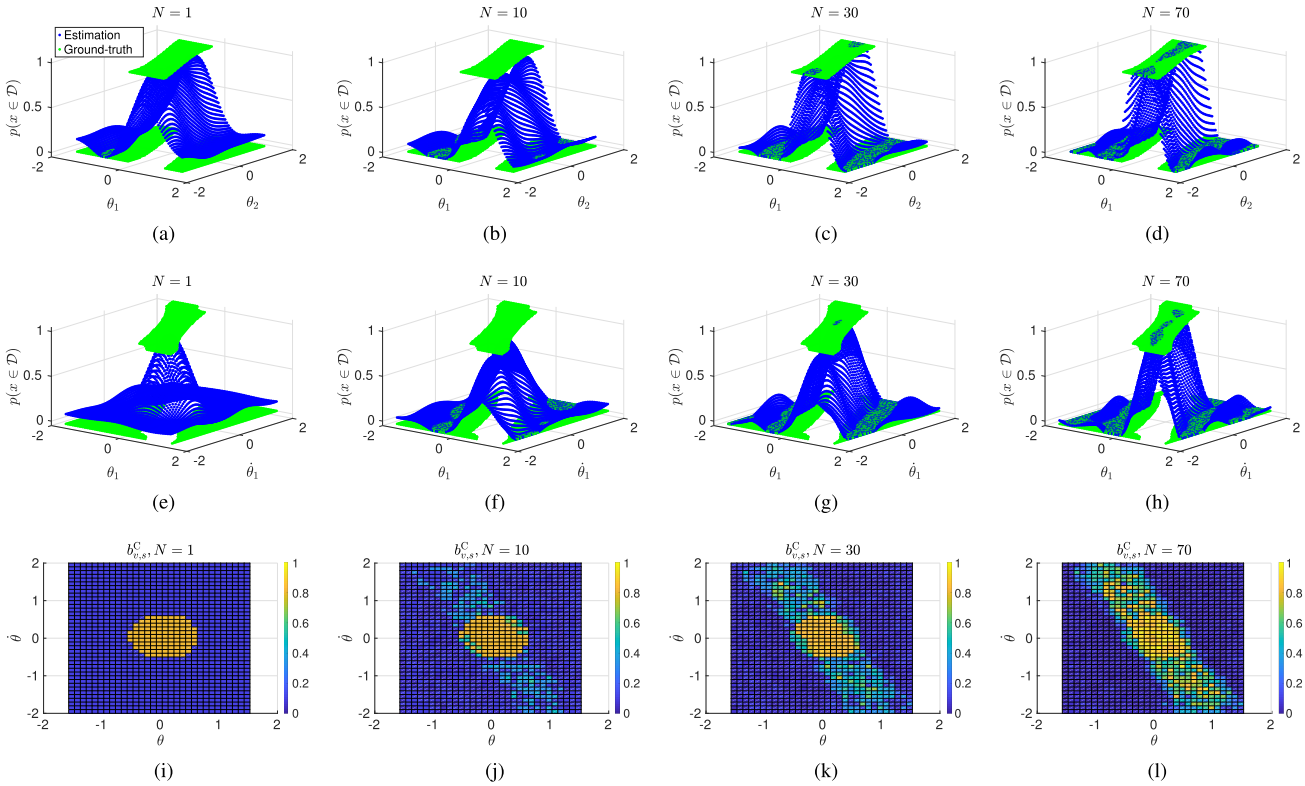


Fig. 7. Online adaptation of the two-link inverted pendulum example. (a)–(d) Probability $p(x \in \mathcal{D})$ from the function approximator $F(x)$ for the slice $\dot{\theta}_1 = \dot{\theta}_2 = 0$ in different supervisor update iterations $N = 1$, $N = 10$, $N = 30$, and $N = 70$. The ground-truth values of $p(x \in \mathcal{D})$, obtained by testing the given corrective controller $K(x)$ at each sampled point, are presented for comparison. (e)–(h) Probability $p(x \in \mathcal{D})$ for the slice $\theta_2 = \dot{\theta}_2 = 0$ in different iterations. (i)–(l) Belief mass $b^{\mathrm{C}}_{v,s}$ from the combined belief map $B^{\mathrm{C}}(v)$ for different index vectors v in different iterations.

iteratively according to the feedback data [see Fig. 7(b), (f), and (j)]. With more iterations, e.g., 30 iterations in Fig. 7(c), (g), and (k), the prediction is quite close to the ground-truth value. Besides, the combined belief map $B^{\mathrm{C}}(v)$ turns out to have a similar shape as the well-known ROA of the one-link inverted pendulum [44]. The performance clearly depends on the amount of feedback data, i.e., as more feedback data are available, not only the prediction, but also the combined belief map $B^{\mathrm{C}}(v)$ becomes more accurate [see Fig. 7(d), (h), and (l)].

The ACC and the FPR of the combined belief map $B^{\mathrm{C}}(v)$ in different update iterations are given in Fig. 8. The threshold for the classification is selected as 0.5. As expected, with more feedback data, the combined belief map $B^{\mathrm{C}}(v)$ gives more accurate estimates while the FPR is kept small. Due to the selected c_1 and c_2 for the feedback belief map $B^{\mathrm{F}}(v)$, several iterations are required until a wrong estimate obtained from the prior belief map $B^{\mathrm{S}}(v)$ is corrected by the feedback data. As a result, the belief about the safe region of the original system $\mathcal{D}$ is expanded cautiously, which leads to the presented slow improvement in the accuracy. The FPR does not reach to 0 because there are some original system states x that, even though, they are mapped to the same grid cell, show different safety properties, such as illustrated in Fig. 4.

Since in this example, the corrective controller $K(x)$ is activated on randomly selected original system states x among the entire state space, more data are required for the update of

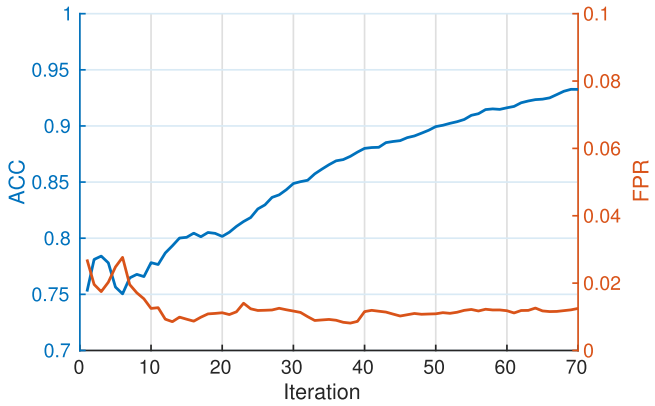


Fig. 8. ACC and FPR of the two-link inverted pendulum example.

the supervisor. However, in practice, we are only interested in locating the decision boundary, as in the example, we described in the next section involving a complex dynamical system.

B. *Quadcopter Flight Control*

To demonstrate the utility and the performance of the proposed safe learning framework for complex dynamical systems, we control the Crazyflie,[1] a lightweight nano quadcopter (see

[1][Online]. Available: https://www.bitcraze.io/crazyflie-2/

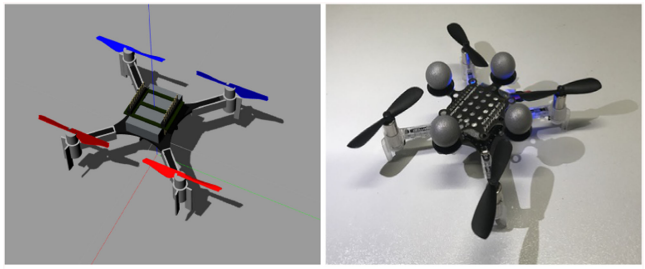


Fig. 9. (a) Crazyflie model in Gazebo simulation environment. (b) Crazyflie with four tracking markers for the Qualisys motion capture system.

Fig. 9). The learning-based controller $\pi(x)$ aims to control the quadcopter to fly while tracking a given constant reference velocity v^d in Cartesian space, and is trained by using the proximal policy optimization (PPO) [45] from OpenAI Baselines.[2] The corrective controller is a PID controller that keeps the quadcopter hovering at a given height. By using the proposed safe learning framework, we reduce the number of quadcopter crashes during the learning process.

The system state of the quadcopter is $x = [p_g, \theta_g, v_b, \omega_b]^T$, 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 in the ground frame, respectively, and $v_b = [v_x, v_y, v_z]^T$ and $\omega_b = [\omega_r, \omega_p, \omega_y]^T$ are the linear and angular velocities in the body frame, respectively. Since positions p_x and p_y have no effect on the corrective controller, the input to the function approximator $F(x)$ that decides on safety of an original system state x has ten dimensions.

For obtaining the simplified system model, we assume that the thrust provided by the motors is large enough, such that the height p_z and the velocity v_z play a less important role in determining whether the current state is safe or not. Therefore, according to the physical properties of the quadcopter, the simplified system state can be chosen as the angular positions θ_r and θ_p and their angular velocities ω_r and ω_p in roll and pitch directions, respectively. However, to be able to visualize the update of the combined belief map $B^{\mathrm{C}}(v)$, we only use the angular velocities ω_r and ω_p as the simplified system state in this example, i.e., $x_s = \Psi(x) = [\omega_r, \omega_p]^T$. Considering the maximal motor speed of the Crazyflie, the simplified system state space is assumed to be in the range of $\omega_r, \omega_p \in [-30\,\text{rad/s}, 30\,\text{rad/s}]$ and is discretized with 1 rad/s. The safe region of the simplified system $\mathcal{D}_s$ is chosen as $\mathcal{D}_s = \{x_s \mid -4 \leq \omega_r \leq 4, -4 \leq \omega_p \leq 4\}$. Note that if angular positions are included to form a more physical meaningful simplified system, SOS programming can be introduced for getting a better initial estimate. An example where SOS programming is used is given in Appendix A.

There are four steps in each learning trial, which are as follows.

1) The quadcopter takes off from the ground and hovers at the position $p_g = [0, 0, 1\ \text{m}]^T$.
2) The learning-based controller starts to control the quadcopter.
3) Once the supervisor suggests that the current state is on the boundary of the estimated safe region, this learning trial is terminated and the corrective controller is activated. Note that in order to provide enough physical space for the corrective controller, the switching also happens if the height of the quadcopter is lower than 0.7 m, i.e., $p_z < 0.7\,\text{m}$. The corrective controller attempts to balance the quadcopter back to a safe hovering state.
4) In the end, if the balance is successful, the quadcopter lands on the ground and waits for the next trial.

The detailed experimental setup can be viewed in the supplementary video.

We examine the performance of the proposed safe learning framework both in simulation and in reality, and present the results as follows.

1) Simulation: The learning process is simulated in Gazebo[3] with the Crazyflie model provided in [46] [see Fig. 9(a)]. The communication between the Gazebo simulator and the PPO algorithm is established through the Robot Operating System (ROS) using the Gym-Gazebo package [47].

To analyze how the supervisor affects the learning process, we perform the simulation in the following different conditions.

1) No supervisor (NS): NS is implemented and the corrective controller is activated only when $p_z < 0.7\,\text{m}$.
2) Feedback data only (FO): The supervisor is trained only with the feedback data. In addition to the height condition, the corrective controller is also activated when $F(x) < \alpha$ with $\alpha = 0.5$ in (11).
3) With supervisor (WS): The proposed framework is implemented and three different thresholds are investigated: $\alpha = 0.2$ (WS$-$0.2), $\alpha = 0.5$ (WS$-$0.5), and $\alpha = 0.8$ (WS$-$0.8).

The learning-based controller $\pi(x)$ controls the four motor speeds and runs at 200 Hz. The parameters as well as the reward function used in this example are given in Appendix B. Starting with a random initial policy, we run the PPO algorithm for 307 200 timesteps (75 updates with 4096 timesteps per each update) and each condition is trained with three different seeds.

The rewards of the learning processes are presented in Fig. 10(a). If a supervisor is used, the reward in the early learning phase is observed to be higher. Since each learning trial is terminated before the system leaves the estimated safe region, the supervisor provides an early-stopping functionality and helps with the policy update. However, more training time is required, because more time is spent on balancing the quadcopter and training the supervisor. The learning performance is affected by the threshold α. While with a low threshold (WS$-$0.2), the supervisor has a minor effect on the learning process, and a high threshold (WS$-$0.8) may lead to an over restricted safe region and, thus, decreases the final learning performance.

The cumulated failures in the first 500 feedback data are given in Fig. 10(b). Since a small threshold allows more risky actions, the threshold α influences the total number of failures as well as the speed of the expansion of the safe region. This can also be observed from the update of the combined belief

[2][Online]. Available: https://github.com/openai/baselines

[3][Online]. Available: http://gazebosim.org/

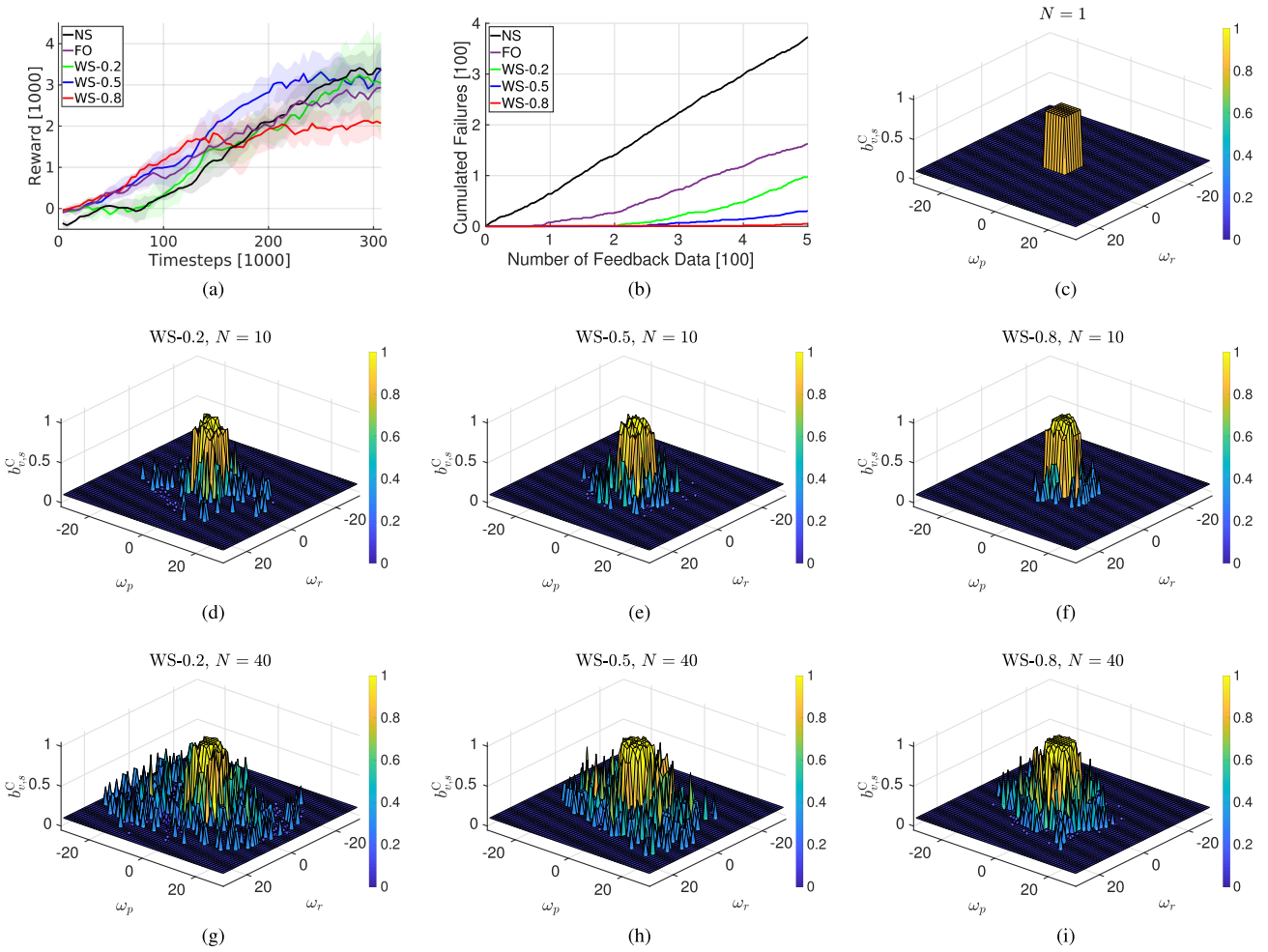


Fig. 10. Simulation results for the quadcopter example. (a) Learning rewards. (b) Corresponding failure counts in the first 500 feedback data. (c) Initialization of the belief mass $b^{\mathrm{C}}_{v,s}$ of the combined belief map $B^{\mathrm{C}}(v)$. (d)–(f) Belief mass $b^{\mathrm{C}}_{v,s}$ for conditions WS$-$0.2, WS$-$0.5, and WS$-$0.8 in the supervisor update iteration $N = 10$. (g)–(i) Belief mass $b^{\mathrm{C}}_{v,s}$ in the supervisor update iteration $N = 40$.

map $B^{\mathrm{C}}(v)$ given in Fig. 10(c)–(i), where a higher threshold leads to a more restricted expansion process. If only feedback data are used (FO), the supervisor is initialized with the same prior belief map, as no feedback data are available prior to the learning process. After the first supervisor update iteration, failures happen immediately as predictions based only on the feedback data are unreliable due to the insufficient data amount. The introduction of the belief maps improves the performance of the supervisor, especially in the early learning phase. The overall success rates of the corrective controller for the entire learning process are: 25% (NS), 58% (FO), 53% (WS$-$0.2), 77% (WS$-$0.5), and 91% (WS$-$0.8). Note that by only using the angular velocities, certain failures cannot be separated, e.g., in Fig. 10(g)–(i), some belief masses $b^{\mathrm{C}}_{v,s}$ are close to 0.5. Therefore, to increase the performance of the combined belief map $B^{\mathrm{C}}(v)$, more features need to be included in the simplified system state.

2) Real-World Experiment: The proposed safe learning framework is also tested for a real Crazyflie [see Fig. 9(b)]. The PPO algorithm runs on a PC with Intel i5-3570 CPU and the corrective controller is implemented on-board. Through the wireless communication provided by the Crazyflie, the learning algorithm receives angular positions θ_g, linear velocities v_b, and angular velocities ω_b from the sensors of the quadcopter. The linear positions p_g are obtained through a Qualisys[4] motion capture system.

Considering the safety of the hardware, the learning process is performed only with supervisor and a moderate threshold $\alpha = 0.5$ (WS$-$0.5). In addition, considering the limited tracking area of the Qualisys system, the corrective controller is also activated if the quadcopter exceeds the region given as $-0.5\,\mathrm{m} \leq p_x \leq 2\,\mathrm{m}$ and $-0.4\,\mathrm{m} \leq p_y \leq 0.4\,\mathrm{m}$. Note that limited by the communication speed between the PC and the Crazyflie, we can only run the learning-based controller at a frequency of 50 Hz. Under such a delay, it is difficult to achieve a stable flight control with motor speeds. To overcome this problem, we select the output u of the learning-based controller as the desired angular positions $\theta^d_g = [\theta^d_r, \theta^d_p, \theta^d_y]^T$ and the desired thrust t^d,

[4][Online]. Available: https://www.qualisys.com/

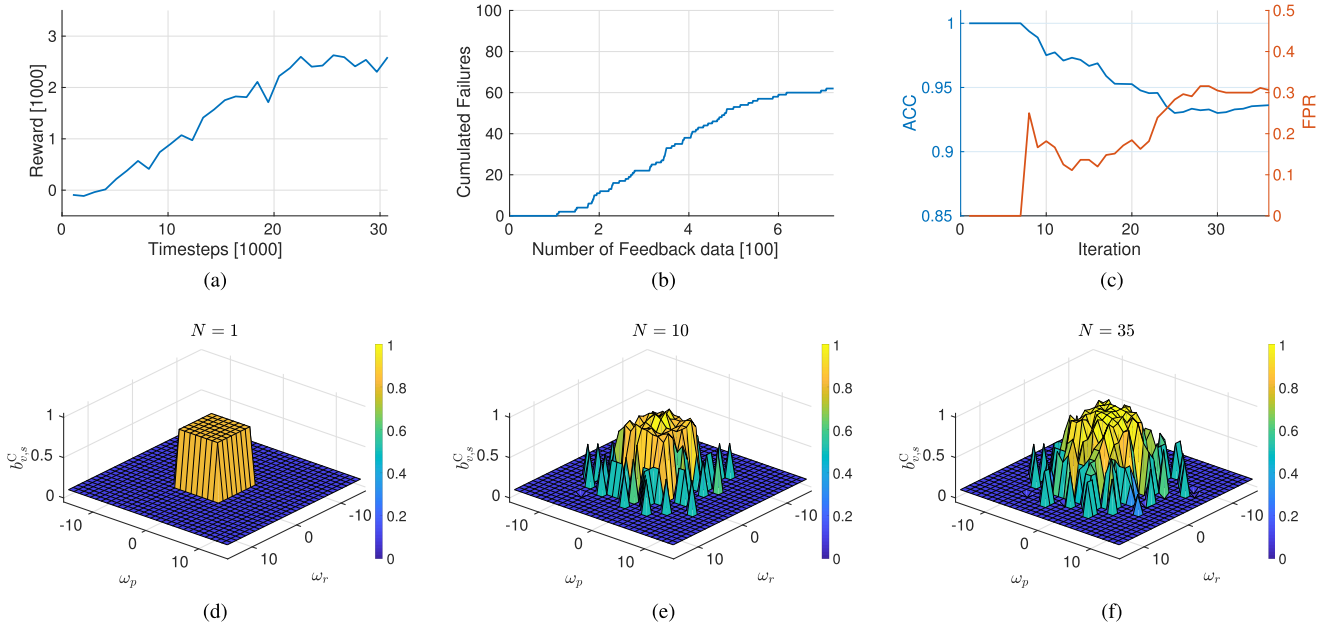


Fig. 11. Results of the quadcopter real-world experiment. (a) and (b) Reward and cumulated failures of the learning process. (c) ACC and FPR of the combined belief map $B^{\mathrm{C}}(v)$. (d)–(f) Belief mass $b^{\mathrm{C}}_{v,s}$ from the combined belief map $B^{\mathrm{C}}(v)$ for different index vectors v in supervisor update iterations $N = 1$, $N = 10$, and $N = 35$, respectively.

i.e., $u = [\theta^d_g, t^d]$. Then, an on-board PID controller that runs at 500 Hz controls the quadcopter to follow the given command. The parameters used in this experiment are given in Appendix B.

Starting with a random initial policy, we perform 30 PPO update iterations (1024 timesteps per each update) of the learning-based controller. Details of the experimental results are presented in the supplementary video. The reward and the accumulated failures are given in Fig. 11(a) and (b), respectively. The success rate of the corrective controller is 89% for the entire learning process. Compared to the simulation, less failures are observed since controlling through desired angular positions and thrust is, in general, safer than directly controlling the motor speeds.

The belief mass $b^{\mathrm{C}}_{v,s}$ of the combined belief map $B^{\mathrm{C}}(v)$ in different supervisor update iterations is shown in Fig. 11(d)–(f). In the early learning phase, due to the conservative estimate of the safe region, the learning-based controller is quickly replaced by the corrective controller. When more feedback data are obtained, the estimate of the safe region is expanded, and therefore, the learning-based controller has more flexibility in choosing its actions. The expansion stops when failures start to happen, which provide information about the boundary of the safe region. The ACC and the FPR are given in Fig. 11(c). Since in the early learning phase, the corrective controller is activated within the initial safe region, the ACC starts from 1. Later, due to the expansion of the safe region, the accuracy decreases as it requires a certain amount of feedback data to compensate the prediction made by the prior belief map $B^{\mathrm{S}}(v)$. Once a more reliable estimate of the safe region is obtained, the accuracy remains high. However, we observe a high FPR in this example. As the simplified system state consists only of angular velocities ω_r and ω_p, it is not precise enough to separate certain failures. For making a better prediction, more information should be included in the simplified system state, e.g., the angular positions θ_r and θ_p.

V. Discussion

In this article, we propose a framework to increase the safety of learning for autonomous systems. A supervisor is constructed to guide the exploration process to prevent the generation of risky behaviors from the intermediate policy. For complex dynamical systems, a simplified system is introduced to enable the practical implementation of the framework. Several critical features of the framework are discussed in this section.

A. Safety in Complex Dynamical Systems

In recent studies [22], [25], the expansion of the safe region is performed based on learning a model of unknown dynamics or disturbances. By executing control commands within the current safe region, these approaches try to predict how the system will behave if it is outside the region. Obtaining such a prediction relies on the assumption that the unknown part follows a certain distribution, e.g., different kernel functions in the Gaussian process represent different characteristics. However, for high-dimensional systems, not only providing suitable assumptions about the distribution is nontrivial, but also a considerable amount of data is required until an accurate model can be learned. Moreover, predicting the system behavior based on complex dynamics is also computational difficult. Therefore, it is challenging to implement such an expansion on complex dynamical systems.

Since both the direct computation of the safe region and the estimation on the complex dynamics pose feasibility challenges, the real safety boundary can effectively be determined only by visiting it. While a prior belief about the safe region is constructed to provide baselines for safety estimates, the decision boundary is modified by using feedback data. With suitable c_1 and c_2 in (23), the decision boundary is expanded cautiously, i.e., only when enough positive feedback data are observed. However, we have to relax the absolute safety guarantee as it is unavoidable that failure will happen when the supervisor tries to learn the real boundary. In that regard, we formulate our framework with the purpose that it can effectively learn from failures, so that in a later learning process, similar dangerous maneuvers are avoided.

B. Safety and Learning Performance

In general, the exploration process of the learning algorithm needs to be restricted to ensure safety, but meanwhile, too much constraining may lead to a poor learning performance. Thus, one central problem of designing a well-performed safe learning approach is to find a suitable balance between maintaining the safety and maximizing the learning performance. In the proposed approach, such a balance is provided by the supervisor decision threshold α, which determines the aggressiveness of actions. An appropriate threshold α not only results in a satisfying learning performance with less failures, but also enables efficient policy update in the early learning process. Since in most tasks, the desired policy should fulfill the safety constraints of the system, the supervisor provides an early-stopping functionality to the learning algorithm by preventing dangerous behaviors.

Finding a good balance between the supervisor and the learning algorithm requires prior knowledge about the system and is usually task dependent. Two measures may assist in relaxing this issue. First, by improving the capability of the corrective controller, a larger safe region can be acquired, which reduces the conflicts between safety and learning performance. Second, safety can also be incorporated in the reward function of the learning algorithm [48]. By encouraging safe behaviors, the learning-based controller tends to stay within the safe region such that less guidance is needed from the supervisor. However, designing a versatile corrective controller or a well-performed reward function often requires a thorough understanding of the task as well as the system.

C. Applications

The supervisory control strategy is compatible with arbitrary reinforcement learning algorithms, thus the proposed safe learning framework is generally applicable to various learning tasks. For example, it can be used to increase safety when a parameterized model-based controller updates its parameters through a learning algorithm [49]. Note that in this work, we treat safety as stability and consider no state constraints. For scenarios where environmental constraints are critical, e.g., collision avoidance, the definition of the safe region has to be modified. For example, control Lyapunov-barrier function [50] can be used in such cases to incorporate state constraints along with stability. The applicability of the proposed framework can be increased with an appropriate description of the safe region. Nevertheless, finding such a description might not be trivial in complicated learning tasks.

Moreover, when applying learning algorithms in real-world scenarios, the learning efficiency is limited by various practical factors, e.g., resetting the environment of the learning algorithm. As demonstrated in the real-world quadcopter experiment, we have to manually put the quadcopter back to a fixed starting position to reset the environment, which requires a considerable amount of time. In general, to reduce the total training time, a reasonable initial policy should be provided, especially when the learning-based controller is expected to accomplish complicated tasks. In that case, the proposed framework aims to increase the safety when the learning algorithm is improving the initial policy according to the real system behavior.

D. Limitations

One major limitation of our framework is that in the early learning phase, the supervisor can only learn about unsafe states by actually visiting them. Thus, although the supervisor is able to adjust its predictions based on feedback data, the framework is only applicable to cases where a reasonable amount of failures is tolerable. For extremely safety-critical cases, where even a single failure is not allowed, an absolute safety guarantee has to be given. However, how to impose such a guarantee for complex dynamical systems is still an open research question.

Besides, since each learning iteration is terminated when the system state is outside the safe region, the learner is only able to learn policies that are contained in this safe region. Therefore, the upper limit of the learning performance is restricted by the choice of the corrective controller. A corrective controller that provides a larger safe region is beneficial for searching the optimal policy, but finding such a corrective controller requires more effort.

Furthermore, to reduce the computational cost, we utilize a simplified system and assume that original system states x with the same corresponding simplified system state x_s share similar safety characteristics. Although this enables the proposed framework to be used on complex dynamical systems, the reliability of the safety estimates depends on the representation capacity of the simplified system. In general, from the perspective of model order reduction, there is a tradeoff between preserving information, which usually results in higher dimensions in the simplified system, and the computational cost. Thus, how to efficiently find a suitable simplified system is a challenging task and further investigations are needed.

VI. Conclusion

In this article, we proposed a general safe reinforcement learning framework, which was implementable for complex dynamical systems. A learning-based controller was combined with a corrective controller to ensure that during the exploration process of the learning algorithm, the system remains inside a safe region. We utilized the concept of ROA to describe the safe region. A supervisor was designed to switch the learning-based controller to the corrective controller when the current state

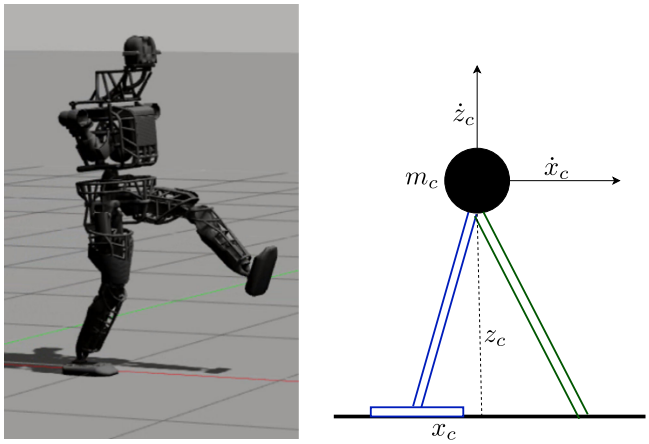


Fig. 12. Humanoid walking example. (a) Atlas model in Gazebo simulation environment. (b) Humanoid robot is simplified to an inverted pendulum model with respect to the CoM.

was on the boundary of the estimated safe region. To enable the practical implementation on complex dynamical systems, a simplified system was introduced to give estimates about the safe region. These estimates were updated online by utilizing the feedback data from the execution results of the corrective controller. Two examples were given to demonstrate the performance of the proposed safe learning framework. A simple two-link inverted pendulum was used to illustrate how the online adaptation method works, whereas a quadcopter control example showed the performance on complex dynamical systems. For future work, one possible direction is to find an effective way to create a representative simplified system from the original system. We believe that the proposed safe learning framework is applicable to a wide range of dynamical systems, and it gives an insight about how to safely extend the modern learning algorithms to real-world tasks.

APPENDIX A
HUMANOID 2-D WALKING

In this example, a humanoid robot is used to further demonstrate the applicability and the performance of the proposed safe learning framework. The Atlas humanoid model (version 1) from DRCSIM[5] is utilized and is constrained to be able to move in the X–Z plane [see Fig. 12(a)]. For simplicity, the arms are excluded from the robot model. Each leg has three motors attached on the hip, the knee, and the ankle joint, which constitute the six motor torque commands. A learning-based controller $\pi(x)$ is trained by the PPO algorithm with the purpose to control the robot to walk forward as fast as possible. By using the proposed safe learning framework, we aim to reduce the possibility of falling to avoid damages to the system. The simulation is performed in Gazebo using OpenAI Baselines and the Gym-Gazebo package.

[5][Online]. Available: https://bitbucket.org/osrf/drcsim/

The state of the humanoid is $x = [p_g, v_g, q^T, \dot{q}^T]^T$, where $p_g = [x_g, z_g, \theta_g]^T$ are the global coordinates and orientation of the body frame with respect to the ground frame and $v_g = [\dot{x}_g, \dot{z}_g, \dot{\theta}_g]^T$ are the body velocities. q and $\dot{q}$ are vectors of six joint angles and six joint velocities, respectively. As x_g and z_g have no effects on determining if the current state can be balanced or not, the input to the function approximator $F(x)$ is 16-D. Based on the CoM m_c of the humanoid, an inverted pendulum is used as the simplified system [see Fig. 12(b)]. The simplified system state is $x_s = [x_c, z_c, \dot{x}_c, \dot{z}_c]^T$, where x_c and z_c are the relative positions of the CoM with respect to the contact point of the support foot, whereas $\dot{x}_c$ and $\dot{z}_c$ are the velocities of the CoM [35]. Considering the physical limits of the system, the CoM properties are assumed to be within the range as $x_c \in [-1\,\text{m}, 1\,\text{m}]$, $z_c \in [0.4\,\text{m}, 1\,\text{m}]$, and $\dot{x}_c, \dot{z}_c \in [-5\,\text{m/s}, 5\,\text{m/s}]$. The simplified system state space is discretized with 0.2 m for x_c and z_c and 1 m/s for $\dot{x}_c$ and $\dot{z}_c$.

The corrective controller $K(x)$ implemented here is a one-step balance controller based on the capture point concept, which defines a point on the ground that the robot can step on to balance itself [51]. Along with the capture point, a reference trajectory for the CoM is generated from the simplified system. The balance controller tries to control the humanoid to step on the capture point while following the CoM reference trajectory. If the trajectory following is not suitable, then the balance controller takes a step with the maximal step length. As a result of stepping functionality for humanoids, the safe region in the safe learning framework is replaced by a concept called N-step viable-capture basin [51]. It represents the set of all initial states from which the robot, with an appropriate control sequence, can come to a stop within n steps (e.g., $n = 0$ means the robot can stabilize itself without stepping). In this example, we use the one-step viable-capture basin as the safe region. It is infeasible to calculate such a basin directly on the original humanoid dynamics, but it can be estimated from the simplified system. The one-step viable capture basin of the inverted pendulum, i.e., the safe region of the simplified system $\mathcal{D}_s$, is obtained by applying the approach described in [44].

The parameters used in this example are given in Appendix B. During the learning, the balance controller is activated if the supervisor believes that the current state is on the boundary of the estimated safe region. By using the feedback data, the belief about the safe region of the humanoid is expanded cautiously until the balance controller starts to fail. Such failures are used to locate the decision boundary for the supervisor. The experimental setup and the resulting behaviors are demonstrated in the supplementary video. The overall success rate of the corrective controller is 79%. Note that while the safety is increased with the proposed safe learning framework, the learning performance is limited by the implemented one-step balance controller. Since the volume of the safe region is restrictive if the humanoid is only allowed to take one step to balance itself. A more satisfying walking behavior can be obtained if a better balance controller is provided. However, improving the balance controller is not the focus of this work and thus is not discussed here.

TABLE I
PARAMETERS OF THE SAFE LEARNING FRAMEWORK

	Inverted Pendulum	Crazyflie Simulation	Crazyflie Real	Humanoid
α	0.5	0.2/0.5/0.8	0.5	0.6
$\overline{p}$	0.8	0.8	0.8	0.8
$\underline{p}$	0.1	0.1	0.1	0.1
σ_0	0.1	0.1	0.1	0.1
c_1	0.2	0.2	0.1	0.3
c_2	0.3	0.5	0.4	0.1
k	1000	8000	8000	10000
k_e	1000	8000	8000	10000
k_u	100	50	20	200
d	0.01	0.1	0.1	0.5
$F(x)$	GPR model: squared exponential kernel	NN: two layers with 128 neurons in each	NN: two layers with 128 neurons in each	NN: two layers with 128 neurons in each

TABLE II
HYPERPARAMETERS OF THE PPO ALGORITHM

Hyperparameter	Crazyflie	Humanoid
Num. steps	4096	8192
Num. epochs	20	20
Num. minibatches	128	256
Adam stepsize	3e-4	1e-4
Discount	0.99	0.99
GAE parameter	0.95	0.95

APPENDIX B
LEARNING PARAMETERS

A. Parameters

The parameters used in this work are given in Tables I and II.

B. Reward Functions

The reward function of the Crazyflie example is

$$r(t) = 1e5||p_t - p_g(t-1)|| - 1e5||p_t - p_g(t)|| \\ -||v_b(t) - v^d||^2 - 0.1||\omega_b(t)||^2$$

where $p_t = 100v^d$ is a virtual target used for giving reward on making progress in the direction of v^d. We use $v^d = [1\text{ m/s}, 0, 0]$ in the given example.

The reward function of the humanoid example is

$$r(t) = 1e4(x_g(t) - x_g(t-1)) - 10{\theta_g}^2 - 2n_j$$

where n_j is the number of joints that are on the limit.

REFERENCES

[1] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[3] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

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

[5] 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, 2017, Art. no. 41.

[6] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[7] 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.*, 2007, pp. 1–8.

[8] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 144–151.

[9] 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.

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

[11] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *Proc. Int. Conf. Learn. Representations Workshop*, 2017.

[12] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[13] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, 2005.

[14] A. Nilim and L. El Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Oper. Res.*, vol. 53, no. 5, pp. 780–798, 2005.

[15] T. M. Moldovan and P. Abbeel, "Safe exploration in Markov decision processes," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1451–1458.

[16] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, vol. 37. Berlin, Germany: Springer, 2012.

[17] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.

[18] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 803–832, 2002.

[19] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. Robot.: Sci. Syst.*, 2016.

[20] 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, 2016.

[21] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[22] 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.

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

[24] 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. Conf. Decis. Control*, 2016, pp. 4661–4666.

[25] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 908–918.

[26] T. Beckers, J. Umlauft, D. Kulic, and S. Hirche, "Stable Gaussian process based tracking control of Lagrangian systems," in *Proc. Conf. Decis. Control*, 2017, pp. 5180–5185.

[27] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, vol. 13. Berlin, Germany: Springer, 2008.

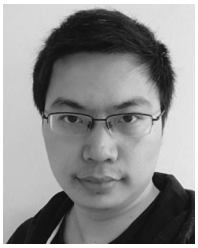
[28] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 8550–8556.

[29] A. Vannelli and M. Vidyasagar, "Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems," *Automatica*, vol. 21, no. 1, pp. 69–80, 1985.

[30] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *Proc. Conf. Decis. Control*, 2003, pp. 4676–4681.

[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, 2019.

[32] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.
[33] M. Sobotka, J. Wolff, and M. Buss, "Invariance controlled balance of legged robots," in *Proc. Eur. Control Conf.*, 2007, pp. 3179–3186.
[34] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2460–2465.
[35] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: Limitations imposed by unilateral contact," in *Proc. Int. Conf. Humanoid Robot.*, 2016, pp. 8–15.
[36] S. Gugercin and A. C. Antoulas, "A survey of model reduction by balanced truncation and some new results," *Int. J. Control*, vol. 77, no. 8, pp. 748–766, 2004.
[37] A. Astolfi and R. Ortega, "Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 48, no. 4, pp. 590–606, Apr. 2003.
[38] G. J. Pappas and S. Sastry, "Towards continuous abstractions of dynamical and control systems," in *Proc. Int. Hybrid Syst. Workshop*, 1996, pp. 329–341.
[39] H. Nickisch and C. E. Rasmussen, "Approximations for binary Gaussian process classification," *J. Mach. Learn. Res.*, vol. 9, no. Oct, pp. 2035–2078, 2008.
[40] D. Kahneman and A. Tversky, "Subjective probability: A judgment of representativeness," *Cogn. Psychol.*, vol. 3, no. 3, pp. 430–454, 1972.
[41] G. Shafer, *A Mathematical Theory of Evidence*, vol. 42. Princeton, NJ, USA: Princeton Univ. Press, 1976.
[42] A. Jøsang, *Subjective Logic*. Berlin, Germany: Springer, 2016.
[43] A. Jøsang, "Categories of belief fusion," *J. Adv. Inf. Fusion*, vol. 13, 2018.
[44] M. Posa, T. Koolen, and R. Tedrake, "Balancing and step recovery capturability via sums-of-squares optimization," in *Proc. Robot.: Sci. Syst.*, 2017, pp. 12–16.
[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347
[46] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular Gazebo MAV simulator framework," in *Robot Operating System*. Berlin, Germany: Springer, 2016, pp. 595–625.
[47] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the OpenAI Gym for robotics: A toolkit for reinforcement learning using ROS and Gazebo," 2016, *arXiv:1608.05742*.
[48] T.-H. Pham, G. De Magistris, and R. Tachibana, "OptLayer—Practical constrained optimization for deep reinforcement learning in the real world," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 6236–6243.
[49] S. R. Friedrich and M. Buss, "A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3365–3372.
[50] M. Z. Romdlony and B. Jayawardhana, "Stabilization with guaranteed safety using control Lyapunov—Barrier function," *Automatica*, vol. 66, pp. 39–47, 2016.
[51] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.



Zhehua Zhou 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. He is currently working toward the Ph.D. degree in learning-based control and robotics with the Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany.

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. degrees in computer science (*summa cum laude*) from Koç University, Istanbul, Turkey, in 2007 and 2010, respectively, and the Ph.D. degree in robotics (*summa cum laude*) from the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany, in 2018.

He is currently a Postdoctoral Researcher with the Machine Learning and Robotics Laboratory, University of Stuttgart, Stuttgart, Germany and Max Planck Institute for Intelligent Systems, Stuttgart, Germany. His research interests include 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 (nee Sobotka) Leibold (Member, IEEE) received the Diploma degree in applied mathematics from the Technische Universität München, Munich, Germany, in 2002, and the Ph.D. degree in motion planning and control of legged robots from the Faculty of Electrical Engineering and Information Technology, Technische Universität München, in 2007.

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

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

In 1988, he was a Research Student for one year with Science University of Tokyo, Tokyo, Japan. From 1994 to 1995, he was a Postdoctoral Researcher with the Department of Systems Engineering, The Australian National University, Canberra, ACT, Australia. From 1995 to 2000, he was a Senior Research Assistant and 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 with 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 been in the Medical Faculty since 2008. Since 2006, he has also been the Coordinator with the Deutsche Forschungsgemeinschaftcluster of excellence "Cognition for Technical Systems (CoTeSys)," Bonn, Germany. His research interests include automatic control, mechatronics, multimodal human system interfaces, optimization, nonlinear, and hybrid discrete-continuous systems.