Technische Universität München
TUM School of Computation, Information and Technology

Safe Reinforcement Learning Methods for Complex Dynamical Systems Based on Model Order Reduction Techniques

Zhehua Zhou

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende/-r: Prof. Dr.-Ing. habil. Dr. h.c. Alexander W. Koch

Prüfende/-r der Dissertation:

1. Prof. Dr.-Ing./Univ. Tokio Martin Buss
2. Prof. Dr.-Ing. Klaus Diepold

Die Dissertation wurde am 31.03.2022 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 17.09.2022 angenommen.

Preamble

This dissertation summarizes my research conducted over the past years at the Chair of Automatic Control Engineering (LSR), at the Technical University of Munich (TUM), Germany. In this chair, I have experienced an excellent and inspiring atmosphere. I would like to take this opportunity to thank all the people who have provided a great help and supported me in achieving my goals.

Foremost, I would like to sincerely thank my advisor and head of the chair Prof. Martin Buss for his valuable support during my years in LSR. His trust and confidence in me encouraged me a lot for completing my research. I have benefited greatly from his insightful comments and suggestions, as well as the freedom he granted in this lab, so that I am able to concentrate on the research topics that I am interested in. I appreciate all his valuable contributions to my work. Furthermore, I would like to thank Dr. Marion Leibold and Dr. Ozgur S. Oguz for their constructive advice and helpful support on my works. The valuable discussions with them have inspired me a lot and made it possible for me to finish my researches.

I thank the whole team at LSR for both the productive and the fun times we shared. A special thanks goes to my friends and colleagues in LSR: Volker Gabler, Khoi Hoang Dinh, Gerold Huber, Rameez Hayat, Stefan Friedrich, Yingwei Du, Markus Schill, Zengjie Zhang, Tim Bruedigung, Tong Liu, Yuhong Chen, Ni Dang, Cong Li, Yongchao Wang, Sebastian Kerz, Salman Bari and Larissa Schimid. Without our extensive discussions, your constructive feedback, and your help during my stay, this work would not have been possible. I would also like to thank Dr. Fangzhou Liu, Dr. Qingchen Liu and Dr. Yi Ren for their constructive suggestions and help with my academic career.

Most importantly, no words can fully express my gratefulness to my family. I would like to thank my parents for their love, continuous support and valuable advice in all these years. Last but not least, I am grateful to all my friends for their support and encouragement during my doctor career.

Abstract

Reinforcement learning is used in many recent studies for solving complicated control tasks, e.g., control of a manipulator or a humanoid. However, although these approaches demonstrate attractive achievements, their results are mostly only presented in simulations. One major concern about applying reinforcement learning methods to real-world dynamical system is safety. Due to the inherent random exploration mechanism, it is unavoidable that the intermediate policy may lead to a dangerous behavior of the system, which is not only harmful to the system itself but also to the environment. Therefore, a safe reinforcement learning method is desired for extending state-of-the-art reinforcement learning algorithms to real-world systems. Moreover, often the motivation of using reinforcement learning for designing the controller is to overcome the computational difficulty in employing traditional controller design techniques. In such cases, the dynamical system usually possesses a complex dynamics, i.e., high-dimensional and highly nonlinear, which hinders the application of many recently proposed safe reinforcement learning approaches. Hence, in this dissertation, we focus on proposing general safe reinforcement learning methods that are suitable for complex dynamical systems.

By using a supervisory control strategy, a safe reinforcement learning framework is realized by switching the actual applied action between the reinforcement learning-based controller and a predefined corrective controller. The central idea is that, through defining a proper control invariant safe region, the learner is able to freely choose its action if the system is inside the safe region. Once the system approaches the boundary of the safe region, the corrective controller is activated for keeping the system safe. For identifying a reliable safe region for complex dynamical systems, we first propose to use a physically inspired model order reduction technique to construct a simplified system model, which then provides estimates about the high-dimensional safe region. Moreover, for having more accurate predictions about safety, the derived safety estimates are updated online by using the observed real system behavior. The results show that, the proposed safe reinforcement learning framework has a satisfying performance in multiple control tasks, and gives an insight about how to safely apply reinforcement learning methods to real-world dynamical systems.

Furthermore, to overcome the limitations of physically inspired model order reduction, we also propose a novel data-driven model order reduction approach for efficiently identifying a low-dimensional representation of the safe region for the safe reinforcement learning framework. This is achieved by learning a representative low-dimensional safety feature from the training data that are obtained by simulating a nominal system model. A modified online adaptation method is also proposed for ensuring the reliability and accuracy of the derived safety estimates. The data-driven approach highly increases the applicability of the safe reinforcement learning framework, and is implementable on a wide range of dynamical systems and learning scenarios.

Finally, considering the fact that the quality of training data significantly affects the performance of data-driven approaches, we present a data generation method for producing training data that are most useful to the safe reinforcement learning framework. The proposed approach results in a training dataset that achieves a satisfying balance between finding an optimal control policy and maintaining the safety of the system, and provides an insight about how different training data will affect the accuracy in predicting safety.

Zusammenfassung

Reinforcement Learning wird in vielen neueren Studien zur Lösung komplizierter Steuerungsaufgaben eingesetzt, z.B. zur Steuerung eines Manipulators oder eines Humanoiden. Obwohl diese Methoden attraktive Leistungen zeigen, werden ihre Ergebnisse jedoch meist nur in Simulationen präsentiert. Ein Hauptanliegen bei der Anwendung von Reinforcement Learning auf reale dynamische Systeme ist die Sicherheit. Aufgrund des inhärenten Zufallsexplorationsmechanismus ist es unvermeidlich, dass die Zwischenpolitik zu einem gefährlichen Verhalten des Systems führen kann, das nicht nur für das System, sondern auch für die Umwelt schädlich ist. Daher ist ein sicheres Reinforcement Learning Methode erwünscht, um moderne Reinforcement Learning Algorithmen auf reale Systeme auszudehnen. Darüber hinaus besteht die Motivation für die Verwendung von Reinforcement Learning zum Entwerfen des Controllers häufig darin, die Rechenschwierigkeiten bei der Verwendung traditioneller Controller-Entwurfsmethoden zu überwinden. In solchen Fällen besitzt das dynamische System normalerweise eine komplexe Dynamik, d.h. hochdimensional und hochgradig nichtlinear, was die Anwendung der vielen vorgeschlagenen sicheren Reinforcement Learning Methoden behindert. Daher konzentrieren wir uns in dieser Dissertation darauf, allgemeine sichere Reinforcement Learning Methoden vorzuschlagen, die für komplexe dynamische Systeme geeignet sind.

Durch die Verwendung einer Überwachungssteuerungsstrategie wird ein sicheres Reinforcement Learning Framework realisiert, indem die tatsächlich angewendete Aktion zwischen dem auf Reinforcement Learning basierenden Controller und einem vordefinierten Korrekturcontroller umgeschaltet wird. Die zentrale Idee ist, dass der Lernende durch die Definition eines geeigneten kontrollinvarianten sicheren Bereichs in der Lage ist, seine Aktion frei zu wählen, wenn sich das System innerhalb des sicheren Bereichs befindet. Sobald sich das System der Grenze des sicheren Bereichs nähert, wird die Korrektursteuerung aktiviert, um das System sicher zu halten. Um einen zuverlässigen sicheren Bereich für komplexe dynamische Systeme zu identifizieren, schlagen wir zunächst vor, eine physikalisch inspirierte Methode zur Reduktion der Modellordnung zu verwenden, um ein vereinfachtes Systemmodell zu konstruieren, das dann Schätzungen über den hochdimensionalen sicheren Bereich liefert. Um genauere Vorhersagen über die Sicherheit zu erhalten, werden die abgeleiteten Sicherheitsschätzungen außerdem online aktualisiert, indem das beobachtete reale Systemverhalten verwendet wird. Die Ergebnisse zeigen, dass das vorgeschlagene sichere Reinforcement Learning Framework eine zufriedenstellende Leistung bei mehreren Steuerungsaufgaben aufweist und einen Einblick darüber gibt, wie man Reinforcement Learning Methoden sicher auf reale dynamische Systeme anwenden kann.

Um die Einschränkungen der physikalisch inspirierten Modellordnungsreduktion zu überwinden, schlagen wir außerdem einen neuartigen datengesteuerten Modellordnungsreduktionsmethode vor, um eine niedrigdimensionale Darstellung der sicheren Bereich für das sichere Reinforcement Learning Framework effizient zu identifizieren. Dies wird erreicht, indem ein repräsentatives niederdimensionales Sicherheitsmerkmal aus den Trainingsdaten gelernt wird, die durch Simulation eines nominellen Systemmodells erhalten werden. Außerdem wird eine modifizierte Online-Anpassungsmethode vorgeschlagen, um die Zuverlässigkeit und Genauigkeit der abgeleiteten Sicherheitsschätzungen sicherzustellen. Die datengesteuerte Methode erhöht die Anwendbarkeit des sicheren Reinforcement Learning Frameworks erheblich und ist auf einer Vielzahl dynamischer Systeme und Lernszenarien umsetzbar.

In Anbetracht der Tatsache, dass die Qualität der Trainingsdaten die Leistung datengesteuerter Methoden erheblich beeinflusst, stellen wir schließlich eine Datengenerierungsmethode zur Erzeugung von Trainingsdaten vor, die für das sichere Reinforcement Learning Framework am nützlichsten sind. Die vorgeschlagene Methode führt zu einem Trainingsdatensatz, der ein zufriedenstellendes Gleichgewicht zwischen der Suche nach einer optimalen Steuerungsrichtlinie und der Aufrechterhaltung der Sicherheit des Systems erreicht und einen Einblick darüber gibt, wie sich unterschiedliche Trainingsdaten auf die Genauigkeit bei der Vorhersage der Sicherheit auswirken.

Contents

Notation

Acronyms and Abbreviations

CoM	center of mass
MOR	model order reduction
ROA	region of attraction
SRL	safe reinforcement learning
TD	temporal difference
SOS	sum-of-squares
GPR	Gaussian process regression
NN	neural network
BBA	basic belief assignment
PPO	proximal policy optimization
SAF	safety assessment function
DSAF	discretized safety assessment function
t-SNE	t-distributed stochastic neighbor embedding
DTW	dynamic time warping

Belief Function Theory

$B(v)$	belief map
$B^{\mathrm{P}}(v)$	prior belief map
$B^{\mathrm{F}}(v)$	feedback belief map
$B^{\mathrm{C}}(v)$	combined belief map
$b_{v,s}$	belief mass of the event $x \in \mathcal{S}$ for index vector v
$b^{\mathrm{P}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ for index vector v in prior belief map

$b^{\mathrm{F}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ for index vector v in feedback belief map
$b^{\mathrm{C}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ for index vector v in combined belief map
b^{i}_{s}	belief mass of the event $x \in \mathcal{S}$ obtained from the i-th training data
$b^{\mathrm{prior}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ in $\mathbb{B}^{\mathrm{prior}}_{v}$
$b^{\mathrm{feedback}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ in $\mathbb{B}^{\mathrm{feedback}}_{v}$
$b^{\mathrm{fuse}}_{v,s}$	belief mass of the event $x \in \mathcal{S}$ in $\mathbb{B}^{\mathrm{fuse}}_{v}$
$b_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ for index vector v
$b^{\mathrm{P}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ for index vector v in prior belief map
$b^{\mathrm{F}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ for index vector v in feedback belief map
$b^{\mathrm{C}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ for index vector v in combined belief map
b^{i}_{u}	belief mass of the event $x \notin \mathcal{S}$ obtained from the i-th training data
$b^{\mathrm{prior}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ in $\mathbb{B}^{\mathrm{prior}}_{v}$
$b^{\mathrm{feedback}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ in $\mathbb{B}^{\mathrm{feedback}}_{v}$
$b^{\mathrm{fuse}}_{v,u}$	belief mass of the event $x \notin \mathcal{S}$ in $\mathbb{B}^{\mathrm{fuse}}_{v}$
σ_v	subjective uncertainty for index vector v
σ^{P}_{v}	subjective uncertainty for index vector v in prior belief map
σ^{F}_{v}	subjective uncertainty for index vector v in feedback belief map
σ^{C}_{v}	subjective uncertainty for index vector v in combined belief map
σ^{i}	subjective uncertainty of the i-th training data
$\sigma^{\mathrm{prior}}_{v}$	subjective uncertainty in $\mathbb{B}^{\mathrm{prior}}_{v}$
$\sigma^{\mathrm{feedback}}_{v}$	subjective uncertainty in $\mathbb{B}^{\mathrm{feedback}}_{v}$
$\sigma^{\mathrm{fuse}}_{v}$	subjective uncertainty in $\mathbb{B}^{\mathrm{fuse}}_{v}$
$\mathbb{B}_v$	BBA for the index vector v
$\mathbb{B}^{i}$	BBA obtained from the i-th training data
$\mathbb{B}_{\mathrm{ini}}$	initial estimate of $\mathbb{B}_v$
$\mathbb{B}^{\mathrm{prior}}_{v}$	prior estimate of $\mathbb{B}_v$
$\mathbb{B}^{\mathrm{feedback}}_{v}$	feedback estimate of $\mathbb{B}_v$
$\mathbb{B}^{\mathrm{fuse}}_{v}$	fused estimate of $\mathbb{B}_v$
$\mathbb{B}_{\varnothing}$	empty BBA

Functions

$f(x)$	function for system dynamics
$g(x)$	function for system dynamics related to input
$d(x)$	unknown and unmodelled part of the system dynamics
$f_s(x)$	function for simplified system dynamics
$g_s(x)$	function for simplified system dynamics related to input
$K(x)$	corrective controller
$\pi(x)$	reinforcement learning-based controller
$f_K(x)$	closed-loop system dynamics w.r.t the corrective controller $K(x)$
$\Psi(x)$	state mapping
$\mathbb{P}(\cdot)$	probability of an event
$\mathbb{E}(\cdot)$	expectation
$F(x)$	function approximator for estimating the safety of x
$L(x)$	locating function
$C_p(v)$	counter of positive feedback data for index vector v
$C_n(v)$	counter of negative feedback data for index vector v
$\Gamma(x_s)$	safety assessment function defined over the simplified state space
$\Gamma_{\mathrm{d}}(v)$	discretized safety assessment function defined over the index vector v
$\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$	prior discretized safety assessment function
$\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$	feedback discretized safety assessment function
$\mathrm{F}(\mathcal{B}_v)$	weighted belief fusion among the set $\mathcal{B}_v$
$P(x)$	function for the probability that system state x has the same safety label both in simulation and reality
$l(x)$	ground-truth labeling function for the safety label of system state x
$h(x)$	hypothesis for the safety label of system state x

Parameters

p_t	probability threshold for the supervisor of the SRL framework
$\overline{p}$	initial probabilistic estimate of safety for states with $\Psi(x) \in \mathcal{S}_s$
$\underline{p}$	initial probabilistic estimate of safety for states with $\Psi(x) \notin \mathcal{S}_s$
σ_0	initial subjective uncertainty for the prior belief map
α	initial value of subjective uncertainty for computing σ_v^{F}
β	decay rate of subjective uncertainty for computing σ_v^{F}
γ	decay rate for updating the subjective uncertainty in $\mathbb{B}^i$
d_t	distance threshold for updating the supervisor
δ	constant for compensating the difference in safety labels
σ_{ini}	initial subjective uncertainty for constructing $\mathbb{B}^i$
σ_{min}	minimum kept subjective uncertainty in $\mathbb{B}^i$
k_{min}	minimum required number of BBAs for performing weighted belief fusion
p_{th}	threshold of p^i_{std} for updating $\mathbb{B}^i$
λ	weighting factor for generating the training dataset

States and Variables

x	n-dimensional original system state
u	m-dimensional control input to the original system
x_s	n_s-dimensional simplified system state
u_s	m_s-dimensional control input to the simplified system
v	index vector of the grid cells in simplified system state space
$s_{\mathrm{sim}}(x)$	safety label of system state x obtained by simulating the nominal system
$s_{\mathrm{real}}(x)$	safety label of system state x obtained from the real system
ω_{ij}	pairwise trajectory distance between the i-th and j-th training data
Ω_{ij}	distance between the i-th and j-th training data
p^i_{mean}	predicted mean value of $P(x^i_{\mathrm{sim}})$ for the i-th training data

p^i_{std}	corresponding standard deviation of p^i_{mean}
$k_{v,s}$	number of safe feedback data for index vector v
$k_{v,u}$	number of unsafe feedback data for index vector v

Sets

$\mathbb{R}$	set of real numbers
$\mathbb{Z}_+$	set of positive integers
$\mathcal{X}$	state space of the original system
$\mathcal{X}_s$	state space of the simplified system
$\mathcal{U}$	action space of the original system
$\mathcal{R}$	ROA of the origin under the corrective controller $K(x)$
$\mathcal{S}$	safe region of the original system
$\mathcal{S}_s$	safe region of the simplified system
$\mathcal{S}_l$	low-dimensional representation of the safe region
$\mathcal{X}_{\text{real}}$	set of feedback data
$\mathcal{X}_{\text{est}}$	set of safety estimates for updating the supervisor
$\mathcal{D}_{\text{train}}$	training dataset
$\mathcal{D}_{\text{feedback}}$	feedback dataset
$\mathcal{B}_v$	set of $\mathbb{B}^i$ for the index vector v
$\mathcal{D}_{\text{ud}}$	sub-dataset generated by using the uniform distribution $\mathcal{N}_{\text{ud}}$
$\mathcal{D}_{\text{mnd}}$	sub-dataset generated by using the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$

Other Symbols

$\Phi(t; x)$	original system trajectory that starts at initial state x when time $t = 0$
$\Phi_s(t; x_s)$	simplified system trajectory that starts at initial state x_s when time $t = 0$
$\Phi_{\text{sim}}(t; x)$	system trajectory in simulation that starts at initial state x when time $t = 0$
$\Phi_{\text{real}}(t; x)$	system trajectory in reality that starts at initial state x when time $t = 0$
D^i_{train}	the i-th training data
D^i_{feedback}	the i-th feedback data
$\mathcal{N}$	distribution of system states of the real system
$\mathcal{N}_n$	distribution of system states of the nominal system contained in the training dataset
$\mathcal{N}_{\text{ud}}$	uniform distribution among the system state space
$\mathcal{N}_{\text{mnd}}$	multivariate normal distribution used for approximating the real distribution $\mathcal{N}$

List of Figures

List of Tables

List of Algorithms

1. Introduction

Reinforcement learning methods are approaches that attempt to solve the common control problem: what an agent should do in different situations [1]. Usually, such a problem includes three aspects: (1) sensation, which means the agent should be able to sense the state of the environment; (2) actions, which are taken by the agent to affect the state; (3) goal, which describes the desired behavior of the agent and is in general represented by a reward function that is related to the states. Through constructing a policy function, reinforcement learning methods map states to actions such that the total reward obtained over the long run is maximized [2]. Like most of the machine learning algorithms, the learner must explore the environment and test the outcomes of different actions in order to find the optimal policy. Reinforcement learning methods often present two characteristics: the trial-and-error policy search process and the delayed rewards, i.e., the actions may not only affect the current reward but also the rewards received in future actions.

The history of reinforcement learning dates back to the early 1950s. At that time, two independent academic disciplines contribute to the formulation of reinforcement learning: optimal control and trial-and-error learning (also referred to as animal learning [3]). While trial-and-error learning is based on psychological concepts and is originated in classical conditioning [4] and instrumental conditioning [5], optimal control solves problems by using the Bellman equation and dynamic programming [6]. These two threads are initially studied independently until the proposition of an influential method called temporal difference (TD) learning in 1970s [7], [8]. TD learning arises from the interdisciplinary study of the two aforementioned threads and hence is used to solve both the trial-and-error learning and optimal control problems [9]. The work of Klopf [10]–[12] is in general considered as the beginning of bringing TD learning methods together with trial-and-error animal learning theories. In 1989, the notable work of Watkins [13], which proposes a method that is well known as Q-learning, fully combines TD learning and optimal control. The integration of these three research directions, trial-and-error learning, optimal control and TD learning, leads to the development of modern reinforcement learning methods.

In the past several decades, thanks to the advances in electrical engineering and computational power, a new concept of reinforcement learning called deep reinforcement learning arises and draws more and more attentions [14]. The name of deep reinforcement learning comes from the fact that it integrates reinforcement learning and deep learning [15], where an artificial neural network that usually contains multiple layers is used to represent the policy, the value or the Q-functions [16] given in the reinforcement learning methods. Since the agent governed by deep neural networks is able to make decisions based on raw inputs [17], deep reinforcement learning provides possibilities of solving complicated control problems without a manual engineering of the state space. Thus, the controller design process could be significantly simplified.

State-of-the-art reinforcement learning or deep reinforcement learning methods have demonstrated attractive achievements in dealing with various tasks, including decision making [18]–[20], image processing [21], [22], natural language processing [23], [24], etc. This encourages the trend of connecting reinforcement learning to traditional control-theoretical disciplines to solve complicated control problems related to dynamical systems [25]. For example, recent studies employ reinforcement learning methods to control humanoids [26], [27], robotic manipulators [28], [29] or autonomous vehicles [30], [31].

However, although these researches show impressive results in controlling dynamical systems with reinforcement learning methods, most of them only perform simulation-based experiments. There still exists a huge gap between simulations and the implementations on real-world dynamical systems. One major impediment for this is that, for real-world systems, safety becomes a significant concern that has to be tackled with in high priority [32]. Note that, from the aspect of control theory, the exact definition of safety could vary depending on the actual task, e.g., from the stability of the controlled system to the avoidance of violations of certain safety constraints. Generally speaking, we consider safety as neither the dynamical system itself nor the environment will be damaged. In traditional reinforcement learning problems, e.g., playing chess or recognizing an image, safety is unfortunately less discussed as it is unimportant or even irrelevant. Hence, if a real-world dynamical system is directly controlled by a standard reinforcement learning method, then due to the inherent trial-and-error learning process, it is very likely that the system or the environment may suffer from severe consequences or damages as the intermediate policies could lead to dangerous behaviours. For example, a quadcopter may crash many times before it learns a successful flying policy.

In order to apply reinforcement learning methods to real-world dynamical systems, a safe reinforcement learning (SRL) method that is able to impose safety guarantees to the learning process is desired. Moreover, often the motivation of using reinforcement learning methods to find the optimal controller is to overcome the computational limitations of traditional model-based controller design techniques [33]. In such cases, the dynamical system is usually highly nonlinear and high-dimensional (referred to as *complex dynamical system* in this dissertation). Hence, considering the practicality, the SRL method should be able to operate on complex dynamical systems. In this dissertation, we therefore focus on proposing general SRL methods that can be effectively used on complex dynamical systems. We believe that the proposed methods are applicable to a wide range of dynamical systems, and they provide a possible solution to the challenging problem of safely extending modern reinforcement learning methods to real-world scenarios.

1.1. Challenges

Depending on the characteristics of the system under consideration, reinforcement learning methods generally can be categorized into two types: for systems that have discrete state and action spaces, and for those with continuous state and action spaces [34]. In this dissertation, we consider SRL methods that are applicable to continuous state and action spaces, as most of the practical dynamical systems that we are interested in belong to this category [35], [36]. For having such a SRL method that is well-performing for complex dynamical systems, following challenges have to be addressed:

(i) How to deal with the highly nonlinear and high-dimensional system dynamics such that the SRL method can be effectively applied to complex dynamical systems.

(ii) In SRL, how to achieve a satisfying balance between finding an optimal control policy with reinforcement learning methods and keeping the system safe.

(iii) If a thorough understanding of the dynamical system is not available, e.g., the system dynamics is partially unknown, how to resolve this problem such that the SRL method is still implementable.

The first challenge comes from the problem that, although there exist multiple researches about SRL, those approaches are not directly applicable to complex dynamical systems due to limitations in computational feasibility. In practical engineering tasks, when the direct operation over the original system is not possible, often a simplified system model is introduced to relax computational burdens [37]–[40]. Inspired by this, we design a SRL framework for complex dynamical systems by using the physically inspired model order reduction (MOR) technique [41] to solve the first challenge.

The second challenge is similar to the well-known exploration and exploitation dilemma in reinforcement learning [42], which states that, on the one hand, the agent must exploit actions that have been already executed before to obtain higher rewards, but on the other hand, it also has to explore unknown actions to make better movements in the future. Similarly, in SRL, providing the learner with more flexibility in action selection is able to result in a better final policy, but meanwhile, this also increases the risk of encountering an unsafe intermediate policy. Moreover, if the learning process is overly restricted for ensuring safety, then the learned final policy may have a poor performance and cannot accomplish the planned control task. To deal with this challenge, we introduce an online adaptation method in the proposed SRL framework that adjusts the estimates about safety by using the observed actual system's behavior. Through a predefined control parameter, the user is able to decide the tendency of the SRL framework between searching for policy and maintaining safety.

The last challenge is a consideration based on the fact that, physically inspired MOR techniques usually require a thorough understanding about the system dynamics [41]. Unfortunately, in many practical scenarios, such an understanding is often not available or difficult to obtain. Under this circumstance, a different way of deriving a simplified system model has to be employed. We assume that at least rough knowledge about the complex dynamical system is known, despite that it could be inaccurate or not complete. This offers us a nominal system model that is able to predict the behaviour of the actual complex dynamical system. Hence, by using a data-driven MOR approach, we propose an extended SRL framework to address the last challenge, which overcomes the limitations of physically inspired MOR. Moreover, as the quality of training data significantly affects the performance of the data-driven MOR, we also present a data generation method that is able to produce representative training data for constructing a well-performed SRL framework.

In the following subsections, we first present an overview about state-of-the-art SRL methods, as well as their limitations. Then, as our approaches are based on the MOR techniques, we give a brief introduction to researches related to this field. In the end of this chapter, we highlight the contributions and outline the structure of this dissertation.

1.1.1. Safe Reinforcement Learning

SRL in Markov decision process

The purpose of SRL is to find an optimal control policy by reinforcement learning while ensuring that certain safety conditions are not violated during the learning process. For systems with discrete action and state spaces, SRL is usually considered in the form of a Markov decision process [43]. In such a case, safety can be realized in two ways: modifying the optimization criterion, or limiting the exploration process.

Often the purpose of modifying the optimization criterion is to include the concept of risk into the learning process. This can be done by, e.g., using the worst case criterion where a policy is considered as optimal if it has the maximum worst-case expected return of the rewards [44]. Besides, an additional risk term can be added to the optimization criterion for balancing the return and the sensitivity to dangerous behaviors [45]. For example, the risk of driving the system to an error state can be included in the expected return to increase safety [46].

Alternatively, the exploration process can be directly controlled to avoid actions that can lead to undesirable or catastrophic situations of the system. However, this usually requires external knowledge about the system and the control task. For instance, initial knowledge gathered from a teacher [47] or a finite set of demonstrations [48] can be incorporated into the learning process for guiding the exploration. Moldovan and Abbeel [49] also proposed a safe exploration algorithm for Markov decision processes, where a constraint about being able to go back to the initial state is imposed. The probabilistic formulation of safety guarantee in [49] inspired later work of using Bayesian optimization as a tool for the safety analysis [50].

Manual control mechanism and transfer learning

SRL in dynamical systems with continuous action and state spaces is in general a more complicated problem and has been a topic of research for over a decade [51]. In earlier works of applying SRL to real-world dynamical systems, safety is often achieved by introducing an additional manual control mechanism. For example in [52], [53], a human pilot takes over the control of the helicopter in case of failures. However, monitoring the entire learning process requires a considerable amount of resources, which limits the applications of these approaches.

It is also worth mentioning that, recent studies that are based on the so called transfer learning [54] concept also attempt to solve the problem of implementing reinforcement learning methods on real-world systems. In these approaches, a satisfying control policy is first trained in the simulator and then transferred to the real dynamical system [55]. However, although transfer learning may reduce the total required learning time on the real system, no safety guarantee has been proposed for the learning process. Due to the inevitable mismatch between simulation and reality (also referred to as the simulation-to-reality gap or reality gap [56]), there exists still a high risk of having an unsafe intermediate policy, especially in the early learning phase on the real-world dynamical systems [57].

Model-free SRL

In model-free scenarios, SRL methods are usually designed by solving a constraint satisfaction problem. For instance in [58], a neural network policy is used to learn a desired

behaviour of the system, while a separate trajectory planning algorithm is employed to select the final behavior and enforce the satisfaction of safety constraints. Besides, constrained policy optimization [59] introduces a constraint to the learning process such that the expected return of cost functions should not exceed certain predefined limits. Alternatively, adding an additional risk term to the reward function, such as risk-sensitive reinforcement learning [60], can also increase the safety of reinforcement learning algorithms. A variety of approaches have been proposed for designing a satisfying risk term in SRL, including variance [61], exponential utility [62], percentile performance [63], etc. However, as no system model is directly considered in these approaches, it is still highly likely that safety conditions are violated during the learning process.

SRL with deterministic safety estimates

When at least an approximated system model is available, a more promising SRL can be realized by combining control-theoretic concepts with reinforcement learning approaches. For example in [64], [65], Lyapunov functions are employed to compute a sub-region of the state space where safety conditions will never be violated. The system is then limited to this sub-region during the learning process. However, finding suitable candidates for Lyapunov functions is challenging if the system dynamics contains uncertainties or is highly nonlinear.

For uncertain dynamical systems, methods based on learning a model of unknown system dynamics [66] or of environmental constraints [67] are proposed to ensure safety during learning. For instance, by predicting the system behavior in the worst case, robust model predictive control [68] is able to provide safety and stability guarantees to reinforcement learning algorithms if the error in the learned model is bounded [69]. Besides, [70] introduces an action governor to correct the applied action when the system is predicted to be unsafe. However, limited by computational efficiency, these approaches with deterministic safety estimates, i.e., the prediction about the safety of a system state is either safe or unsafe, are usually only applicable to linear systems. Moreover, the accuracy of the learned model also strongly affects the performance of these approaches.

SRL with probabilistic safety estimates

To relax demands placed on the accuracy of the system model and extend SRL to nonlinear systems, instead of deterministic safety estimates, recent studies employ probabilistic safety estimates, in which safety predictions are represented as probabilities. For example in [71], modelling uncertainties are approximated by Gaussian process models [72], and a probabilistic safe region is computed by solving the Hamilton-Jacobi-Isaacs equation [73] from the reachability analysis [73]. Similarly, Gaussian process models are used in [74], [75] to model unknown system dynamics. A safe region is then obtained from the probabilistic estimate of the region of attraction (ROA) of a safe equilibrium state. The key component of these studies is a forward invariant safe region, such that the learning algorithm has the flexibility to execute desired actions within the safe region. Safety is ensured by switching to a safety controller whenever the system approaches the boundary of the safe region. However, the safe region is computed either by solving a partial differential equation in [71] or sampling in [75], both of which suffer from the curse of dimensionality [76]. Furthermore, modeling an unknown dynamics or disturbance with Gaussian process models also poses challenges when the system is highly nonlinear and high-dimensional, since both making adequate as-

sumptions about the distribution of dynamics and acquiring a sufficient amount of data are difficult. Therefore, although approaches like [71], [75] enable promising results with low-dimensional dynamical systems[1], they are not directly applicable to complex dynamical systems [77].

In most cases where a learning-based controller is preferred, it is likely that traditional controller design techniques are difficult to be implemented, mainly due to the complex system dynamics [78]. Hence, a desired SRL method should be capable of working on complex dynamical systems. In this dissertation, we thus focus on proposing a novel SRL framework for complex dynamical systems, which overcomes the computational limitations of the aforementioned SRL methods.

1.1.2. Model Order Reduction Techniques

When it is infeasible to perform operations on the complete full system model, MOR techniques are widely used to reduce the computational complexity of the control problem by finding a reduced order model [79]. Such a simplified system model in general has a lower dimensional state space and works as an approximation of the original system. MOR techniques can be categorized into three types: physically inspired, projection-based and data-driven [80]. In this subsection, we present a brief introduction to these approaches.

Physically inspired MOR

Physically inspired MOR techniques find the simplified model by taking into account domain expertise and in-depth knowledge of the implementation details on the original system [41]. The identified simplified system model is able to describe the predominant characteristics of the original system and estimate the output of interest with high enough precision. For example, in control of a humanoid, it is common to use an inverted pendulum model of the center of mass (CoM) for approximating the full system dynamics [81]. Then the controller is designed such that the humanoid's CoM follows a desired trajectory for the CoM that is computed from the simplified system model. However, for applying physically inspired MOR techniques, it often requires a thorough understanding of the dynamics of the controlled system.

Projection-based MOR

Projection-based MOR derives a reduced order model by projecting the original system equations onto a low-dimensional subspace that is constructed to represent the essential character of the system's original input-output relationship [82]. Usually, the state of the original system at selected inputs is used to construct a basis for the low-dimensional subspace with different basis generation methods [83]. The lower-order approximation of the system model is computed by neglecting system states that have relatively low effect on the overall model response. For stable linear time-invariant systems, projection-based MOR approaches, e.g., balanced truncation [38], are able to guarantee a simplified system that is asymptotically

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

stable and has bounded errors for the output of interest [84]. However, limited by the computational cost, it is difficult or impossible to find a reasonable basis for highly nonlinear and high-dimensional system dynamics, which hinders the application of projection-based MOR on complex dynamical systems.

Data-driven MOR

Data-driven MOR techniques are learning-based and use data obtained from the original system that represent the input-output relation for deriving a lower dimensional system model [85]. In general, the obtained simplified model uses combinations of different basis functions to approximate the input-output relationship of the original system. For having an accurate approximation, interpolation or regression methods are employed to adjust the coefficients of the basis functions such that their combinations best reproduce the mapping between the inputs and the outputs given by the training data. Depending on the actual task, different forms of basis functions can be used, e.g., polynomials or radial basis functions [86]. The type of basis functions highly affects the quality of the identified simplified model. Due to the high flexibility and applicability, impressive results of data-driven MOR are presented in various scenarios and tasks, e.g., using support vector machines [87] or autoencoder [88] in machine learning tasks. In this dissertation, we also employ data-driven MOR for designing a more flexible SRL framework that is able to overcome the limitations of physically inspired MOR techniques.

1.2. Contributions and Outline

In this dissertation, we provide a possible solution to the challenging problem how to safely and efficiently apply state-of-the-art reinforcement learning methods to complex dynamical systems. In Chapter 2, we first introduce preliminary knowledge about how reinforcement learning methods are applied to dynamical systems with continuous state and action spaces, as well as a SRL framework for low-dimensional dynamical systems that is based on a supervisory control strategy. Then in Chapter 3, we present a general SRL framework for complex dynamical systems that utilizes physically inspired MOR to find a simplified system model. Such a low-dimensional system model approximates the behaviour of the original system and predicts whether the original system is safe or not in its current state. Besides, we also discuss in Chapter 3 how to achieve a satisfying balance between finding an optimal policy and maintaining the safety by introducing an online adaptation method in the proposed SRL framework. In Chapter 4, we explain in detail how to extend the SRL framework with a data-driven MOR approach, such that the limitations of physically inspired MOR are addressed. The data-driven MOR enables the application of the proposed SRL framework to a wider range of dynamical systems and control scenarios. Moreover, in Chapter 5, we introduce a data generation method that is able to produce representative training data for the data-driven MOR. By using the proposed approach, we realize a SRL framework that achieves a better performance in the early learning phase compared to the cases where traditional data generation methods are used. Finally, Chapter 6 concludes this dissertation and gives potential future work on the topic of SRL.

The contributions of this dissertation are summarized as follows:

(i) **Proposition of a SRL framework for complex dynamical systems (Chapter 3)**

We propose a generalizable SRL framework for complex dynamical systems that allows autonomous systems to learn in the real-world safer than standard reinforcement learning methods. With a proper definition of the safe region, the learner is able to freely select its actions as long as the system is inside the safe region. A predefined corrective controller is activated to keep the system safe once it reaches the boundary of the safe region. By utilizing such a supervisory control strategy, the framework is compatible with arbitrary reinforcement learning algorithms, and is also applicable to various learning scenarios. For dealing with complex system dynamics, we utilize physically inspired MOR to construct a simplified system model, which facilitates an estimation for the safe region and leads to a good initialization of the SRL framework. During the learning process, the safe region is modified through an online adaptation method such that the flexibility of the learning-based controller is increased. The proposed SRL framework enables applications of state-of-the-art reinforcement learning algorithms in real-world scenarios, and provides practical insights on how to achieve a good balance between safety and learning performance.

The method and results presented in Chapter 3 have been published in [89].

(ii) **Proposition of a data-driven approach for SRL (Chapter 4)**

To overcome the limitations of physically inspired MOR, we propose a novel data-driven approach to identify the simplified system model. Inspired by transfer learning [90], we assume that an approximated system model of the complex dynamical system is available. By simulating this approximated model, we obtain training data that represent safety of various original system states. Then, a data-driven approach that computes probabilistic similarities between each training data is proposed to learn a low-dimensional representation of the safe region. Such a low-dimensional safe region is used as the starting point to SRL in the real system. For having more reliable safety estimates, we also introduce a modified online adaptation method to account for the inaccuracy in training data. The proposed data-driven approach is capable of systematically identifying a low-dimensional representation of the safe region and is implementable on a wide range of dynamical systems and control tasks, which highly increases the applicability of the SRL framework.

The method and results presented in Chapter 4 have been published in [91].

(iii) **Proposition of a data generation method for data-driven SRL (Chapter 5)**

For the SRL framework where data-driven MOR is used to construct the simplified system model, the quality of training data significantly affects the performance of the identified low-dimensional safety feature. To improve the reliability of the SRL framework, especially in the early learning stage where only limited feedback data is available, we introduce a data generation method for producing representative training data. This is achieved by first analyzing the factors that influence the generalization

error of the obtained safety estimates, i.e., the error in predicting safety of the real system by using information from the simulated nominal system. Then, a data generation method that combines a uniform distribution and a multivariate normal distribution is designed accordingly. By adjusting the weights of these two distributions, the proposed approach realizes a satisfying balance between finding an optimal control policy and keeping the system safe for the SRL framework. Moreover, taking the used data-driven MOR as an example, we provide an insight about what could be a useful way to generate training data when any other data-driven method is employed to predict the safety of a dynamical system.

The method and results presented in Chapter 5 have been published in a preprint [92].

Reinforcement Learning in Dynamical Systems

2.

Reinforcement learning offers a framework and set of tools for the design of complicated and hard-to-engineer behaviors of the dynamical systems. Instead of explicitly solving the control problem, reinforcement learning finds the optimal controller via trial-and-error interactions with the environment. Unlike most of the reinforcement learning problems discussed in machine learning community, controlling a dynamical system is often a problem related to continuous state and action spaces, as well as high-dimensional state space.

In this chapter, we first present a brief introduction to the components of a reinforcement learning problem in Section 2.1. Then, details about how reinforcement learning methods solve the control problem of a dynamical system are given in Section 2.2. In Section 2.3, a SRL framework that is directly implementable on low-dimensional dynamical system is explained.

2.1. Components of Reinforcement Learning Problem

Reinforcement learning problem is often formulated through the interaction between a learning agent and the environment. While the agent attempts to maximize the accumulated reward over a long run, the environment reacts to the actions of the agent and results in a new system state as well as a scalar reward for describing the performance of the applied action. In such reinforcement learning problems, the status of the agent and the environment is represented as a state $s \in S$, and the agent is able to perform an action $a \in A$ to affect the state. In this dissertation, we consider both the state space S and the action space A as continuous and multi-dimensional.

A state s contains all relevant information about current system's situation and is equivalent to observation in control problems of a dynamical system. An action a is the control signal applied to the dynamical system. To describe the influence of an action a on the state s, a transition probability $T(s^{'}, a, s) = \mathbb{P}(s^{'}|s, a)$ is introduced to capture the dynamics of the system. For every control step, the agent also receives a scalar reward r from the environment, which is assumed to be a function of the state s and the action a, i.e., $r = r(s, a)$. Hence, the goal of reinforcement learning is to find a mapping, often referred to as a policy π, that selects an action according to the current state such that the cumulative expected reward is maximized. The policy π can be either deterministic or probabilistic, where the former uses the exact same action a for a given state s, i.e., $a = \pi(x)$, and the later draws a sample from a distribution over actions, i.e, $a \sim \pi(s, a) = \mathbb{P}(a|s)$. The fundamental problem in reinforcement learning is to discover the relations between states, actions and rewards. To this purpose, exploration in the state-action space is required, which can either be directly embedded in the policy or performed separately and only as part of the learning process.

In general, reinforcement learning methods search for an optimal policy π^* that maximizes the expected return J. However, based on the actual learning task, there exist different definitions of the expected return J that lead to different optimal behaviors of the dynamical system [93].

The most straightforward model of expected return is the finite-horizon model given as

$$J = \mathbb{E}\left\{\sum_{h=0}^{H} r_h\right\} \tag{2.1}$$

where H defines the length of the horizon. As only the rewards in next H timesteps are considered in such a formulation, it is suitable to control problems where how many steps are remaining is known. Alternatively, future rewards can be discounted by a discount factor ν that leads to the following expected return

$$J = \mathbb{E}\left\{\sum_{h=0}^{\infty} \nu^h r_h\right\} \tag{2.2}$$

where it holds that $0 \leq \nu < 1$. Such a formulation is most frequently discussed in reinforcement learning community and the discount factor ν qualitatively changes the form of the optimal policy. Policies designed by optimizing with small discount factor ν are myopic and greedy, and may lead to a poor performance if longer term rewards are important. Besides, for dynamical systems, the optimal control law can be unstable if the discount factor ν is too low. Therefore, discounted expected return is often not well suited for controlling a dynamical system. When the discount factor ν approaches 1, the discounted expected return (2.2) becomes what is known as the average expected return

$$J = \lim_{H\to\infty} \mathbb{E}\left\{\frac{1}{H}\sum_{h=0}^{H} r_h\right\} \tag{2.3}$$

This formulation has the problem that it cannot distinguish between policies that initially gain a transient of large rewards and those that do not. However, in real-world dynamical systems, the drawbacks of the discounted expected return are often more critical than those of the average expected return, as stable behavior is often more important than a good transient. In next subsection, we explain how reinforcement learning methods solve the optimization problem of maximizing the expected return by using the average expected return as an example. Similar results for finite-horizon and discounted expected return can be found in many reinforcement learning literature, e.g. [1].

2.2. Value Function Based Approaches and Policy Search

To enable the optimization in continuous state and action spaces, we denote the policy as a conditional probability distribution $\pi(s, a) = \mathbb{P}(a|s)$, and consider it as parameterized by a vector θ. By incorporating the policy, the average expected return is represented as

$$J(\pi) = \sum_{s,a} \mu^\pi(s)\pi(s, a)r(s, a) \tag{2.4}$$

where μ^π is the stationary state distribution generated by policy π with respect to the system dynamics. The goal of reinforcement learning is hence to find an optimal policy π^* (or equivalent policy parameters θ^*) that maximizes (2.4). Such a control problem can be framed as the following optimization problem

$$\begin{aligned} \max_\pi J(\pi) &= \sum_{s,a} \mu^\pi(s)\pi(s,a)r(s,a), && (2.5)\\ \text{s.t. } \mu^\pi(s') &= \sum_{s,a} \mu^\pi(s)\pi(s,a)T(s',a,s), \forall s' \in S, && (2.6)\\ 1 &= \sum_{s,a} \mu^\pi(s)\pi(s,a) && (2.7)\\ \pi(s,a) &\geq 0, \forall s \in S, a \in A && (2.8) \end{aligned}$$

where (2.6) defines stationarity of the state distributions μ^π and (2.7) ensures a proper state-action probability distribution. This optimization problem can be solved in two different ways: either optimize in the Lagrange dual formulation of (2.5)-(2.8), or search the optimal solution directly in the original problem. The former is known as value function based approach, and the latter is referred to as policy search in reinforcement learning.

2.2.1. Value Function Based Approaches

For solving the optimization problem (2.5)-(2.8) in the dual form, we use Lagrange multipliers $V^\pi(s')$ and $\bar{R}$ to express the Lagrangian of the problem as

$$\begin{aligned} L &= \sum_{s,a} \mu^\pi(s)\pi(s,a)r(s,a)\\ &+ \sum_{s'} V^\pi(s') \left[\sum_{s,a} \mu^\pi(s)\pi(s,a)T(s',a,s) - \mu^\pi(s')\right] + \bar{R}\left[1 - \sum_{s,a} \mu^\pi(s)\pi(s,a)\right]\\ &= \sum_{s,a} \mu^\pi(s)\pi(s,a)\left[r(s,a) + \sum_{s'} V^\pi(s')T(s',a,s) - \bar{R}\right]\\ &- \sum_{s'} V^\pi(s')\mu^\pi(s') \underbrace{\sum_{a'} \pi(s',a')}_{=1} + \bar{R} \end{aligned} \quad (2.9)$$

Using the property $\sum_{s',a'} V(s')\mu^\pi(s')\pi(s',a') = \sum_{s,a} V(s)\mu^\pi(s)\pi(s,a)$, we can obtain the Karush-Kuhn-Tucker conditions by differentiating with respect to $\mu^\pi(s)\pi(s,a)$, which yields extrema at

$$\frac{\partial L}{\partial \mu^\pi \pi} = r(s,a) + \sum_{s'} V^\pi(s')T(s',a,s) - \bar{R} - V^\pi(s) = 0 \quad (2.10)$$

This implies that there are as many equations as the number of states multiplied by the number of actions. For each state there can be one or several optimal actions a^* that result in the same maximal value. The optimal value function can be written in terms of the optimal action a^* as $V^{\pi^*}(s) = r(s,a^*) - \bar{R} + \sum_{s'} V^{\pi^*}(s')T(s',a^*,s)$. As a^* is generated by the same optimal policy π^*, we know the condition for the multipliers at optimality is

$$V^{\pi^*}(s) = \max_{a^*}\left[r(s,a^*) - \bar{R} + \sum_{s'} V^{\pi^*}(s')T(s',a^*,s)\right] \quad (2.11)$$

Such a statement is equivalent to the Bellman principle of optimality [6]. Thus, we have to perform an optimal action a^* and follow the optimal policy π^* in order to achieve a global optimum. When evaluating (2.11), the optimal value function $V^{\pi^*}(s)$ corresponds to the long term additional reward, beyond the average reward $\bar{R}$, gained by starting in state s while taking optimal actions a^* according to the optimal policy π^*.

Reinforcement learning approaches that are based on identifying solutions to the aforementioned dual formulation of the optimization problem are known as value function based approaches. Instead of directly learning a policy, these approaches first approximate the Lagrangian multipliers $V^{\pi^*}(s)$ and use it to reconstruct the optimal policy. The value function $V^{\pi}(s)$ is defined equivalently, however instead of always taking the optimal action a^*, the action a is picked according to a policy π

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \left(r(s,a) - \bar{R} + \sum_{s'} V^{\pi}(s') T(s', a, s) \right) \tag{2.12}$$

For increasing the computational efficiency, many reinforcement learning methods also utilize the state-action value function $Q^{\pi}(s,a)$ that is defined as

$$Q^{\pi}(s,a) = r(s,a) - \bar{R} + \sum_{s'} V^{\pi}(s') T(s', a, s) \tag{2.13}$$

which, compared to the value function $V^{\pi}(s)$, explicitly contains the information about the effects of a specific action. The optimal state-action value function is

$$\begin{aligned} Q^{\pi^*}(s,a) &= r(s,a) - \bar{R} + \sum_{s'} V^{\pi^*}(s') T(s', a, s) \\ &= r(s,a) - \bar{R} + \sum_{s'} \left(\max_{a'} Q^{\pi^*}(s', a') \right) T(s', a, s) \end{aligned} \tag{2.14}$$

It can be shown that an optimal and deterministic policy $\pi^*(s)$ can be reconstructed by always picking the action a^* in the current state that leads to the state s with the highest value $V^{\pi^*}(s)$

$$\pi^*(s) = \arg\max_{a} \left(r(s,a) - \bar{R} + \sum_{s'} V^{\pi^*}(s') T(s', a, s) \right) \tag{2.15}$$

If the optimal value function $V^{\pi^*}(s')$ and the transition probabilities $T(s', a, s)$ are known, determining the optimal policy is straightforward for systems with discrete actions as an exhaustive search is possible. However for continuous action spaces, determining the optimal action a^* is an optimization problem in itself. Function approximation is often employed in this case to find a low-dimensional representation that matches the real value function. If the state-action value function $Q^{\pi^*}(s,a)$ is used instead of the value function $V^{\pi^*}(s)$

$$\pi^*(s) = \arg\max_{a} \left(Q^{\pi^*}(s,a) \right) \tag{2.16}$$

then the calculation of the weighted sum over the successor states is avoided. Thus no knowledge of the transition function is required.

A wide variety of value function based approaches have been developed to estimate the optimal value function $V^{\pi^*}(s)$ or the state-action value function $Q^{\pi^*}(s,a)$, including dynamic programming based approaches like policy iteration or value iteration [94], rollout based Monte Carlo approaches [95], and TD methods [93]. The readers may refer to further literature in reinforcement learning for more details about these approaches.

2.2.2. Policy Search

In contrast to value function based approaches, the policy search finds the optimal policy directly on the original optimization problem. It allows a natural integration of expert knowledge, e.g., through initialization of the policy, to the controller design process, and is usually computational efficient as optimal policies in general have many fewer parameters than optimal value functions. In control problems of a dynamical system, policy search approaches have become an important alternative to value function based approaches due to a better scalability.

Most of the policy search approaches optimize locally around an existing policy π, which is parameterized by a set of policy parameters θ_i, by computing changes in the policy parameters $\Delta\theta_i$ that will increase the expected return. This results in iterative updates of the policy in the following form

$$\theta_{i+1} = \theta_i + \Delta\theta_i \tag{2.17}$$

The key computation here is the determination of the policy change and a variety of approaches have been proposed to solve this problem. These approaches can be broadly categorized into "black box" and "white box" methods. Black box methods are general stochastic optimization algorithms that use only the expected return of policies that is estimated by sampling. These approaches leverage neither knowledge about the system nor the structure of the control problem. While white box methods take advantage of some of additional information within the reinforcement learning domain, e.g. the system dynamics.

One of the popular policy search approaches that is well suited for dynamical systems is the gradient based approach. The updates of the policy parameters are based on a hill-climbing approach that is following the gradient of the expected return J for a defined step size ρ

$$\theta_{i+1} = \theta_i + \rho\nabla_\theta J \tag{2.18}$$

There exist many different methods for estimating the gradient $\nabla_\theta J$, e.g., finite difference gradients method [96], likelihood ratio method [97], etc. However, most of theses approaches require tuning of the step size ρ.

As an alternative to gradient based approaches, approaches that are inspired by expectation-maximization, e.g. Monte Carlo expectation-maximization [98], or dynamic programming, e.g. differential dynamic programming method [99], have also been shown to be successful for dynamical systems. However, how to choose an appropriate method that best solves the given control task is still an open and challenging research problem, and a satisfying decision often requires preliminary knowledge about the characteristic and structure of the control problem.

2.2.3. Actor-critic Method

Value function based approaches are in general difficult to be applied to high-dimensional dynamical systems, as they require a function approximation of the value function for finding the optimal policy. With high-dimensional state and action spaces, most of the theoretical guarantees in these approaches become invalid and even finding the optimal action can be a challenging problem due to the brittleness of the approximation and the high computational cost. Moreover, in value function based approaches, the largest local error determines the quality of the resulting policy. This results in a error propagation problem in value functions,

i.e., a small change in the policy may result in a large change in the value function, which again causes a large change in the policy. Such a learning process often leads to an unstable behaviour under the function approximation and is dangerous if applied to a real-world dynamical system.

In contrast, policy search approaches usually only consider the current policy and its neighborhood in order to gradually improve the performance. Although in most cases only local optimal solutions are obtained, these approaches work well in conjunction with continuous features. Besides, local coverage and local errors also result into improved scalability in dynamical systems.

Policy search and value function based approaches are also referred to as actor-only and critic-only methods, respectively. The idea of a critic is to first observe and estimate the performance of the applied controller, and then derive a policy based on the gained knowledge. Conversely, the actor directly tries to find the optimal policy. Based on this, a set of algorithms called actor-critic methods attempts to incorporate the advantages of both policy search and value function based approaches. In these approaches, the policy and the value function are explicitly maintained. The value function, i.e., the critic, is not employed for action selection. Instead, it observes the performance of the actor and decides when the policy needs to be updated and which action should be preferred. The resulting update process features the local convergence properties of policy gradient algorithms while reducing the update variance. Note that, there is a trade-off between the benefit of reducing the variance and having to learn a value function, as the samples required to estimate the value function could also be employed to obtain better gradient estimates for the policy update.

2.3. SRL Based on ROA and Supervisory Control

For safely applying reinforcement learning methods on dynamical systems, we explain a SRL framework that is based on the usage of a supervisory control strategy in this section. The purpose of the SRL framework is to learn a control policy with reinforcement learning while satisfying certain safety constraints during the learning process. From the control theoretical perspective, one major criterion related to the safety of a dynamical system is stability. Therefore, we consider stability as the safety condition in this dissertation. In this regard, if a given equilibrium state is known to be a safe state and is locally asymptotically stable under a given control policy, then the ROA of this equilibrium state forms a safe region of the system. A supervisory control scheme, which switches between the learning-based controller and a predefined corrective controller that attempts to drive the system back to the safe equilibrium state, is constructed based on the safe region, such that the system remains safe during the training procedure. Details about the SRL framework are presented in the remaining part of this section.

2.3.1. System Model and ROA

A general nonlinear control affine dynamical system is given by

$$\dot{x} = f(x) + g(x)u, \tag{2.19}$$

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

$$\dot{x} = f_K(x), \tag{2.20}$$

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 dissertation, 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

$$\mathcal{R} = \{x_0 \in \mathcal{X} \mid \lim_{t\to\infty} \Phi(t; x_0) = 0\}, \tag{2.21}$$

where $\Phi(t; x_0)$ denotes the system trajectory of (2.20) 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 as follows.

Definition 1. *A closed positive invariant subset of the ROA $\mathcal{R}$ containing the origin is defined as the safe region $\mathcal{S}$ of the closed-loop dynamical system (2.20).*

2.3.2. SRL Framework

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

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

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

$$u = \begin{cases} \pi(x), & \text{if } t < t^*, \\ K(x), & \text{else,} \end{cases} \tag{2.22}$$

where t^* is the first time point where the system state x is on the boundary of the safe region $\partial\mathcal{S}$. For each learning iteration, the system state x starts inside the safe region $\mathcal{S}$ for time $t = 0$. The learning algorithm has the flexibility to evolve the trajectory within the safe region $\mathcal{S}$ and update the control policy $\pi(x)$. Once the system state x is on the boundary of the safe region, i.e., $x \in \partial\mathcal{S}$, this learning iteration is terminated at time $t = t^*$. The

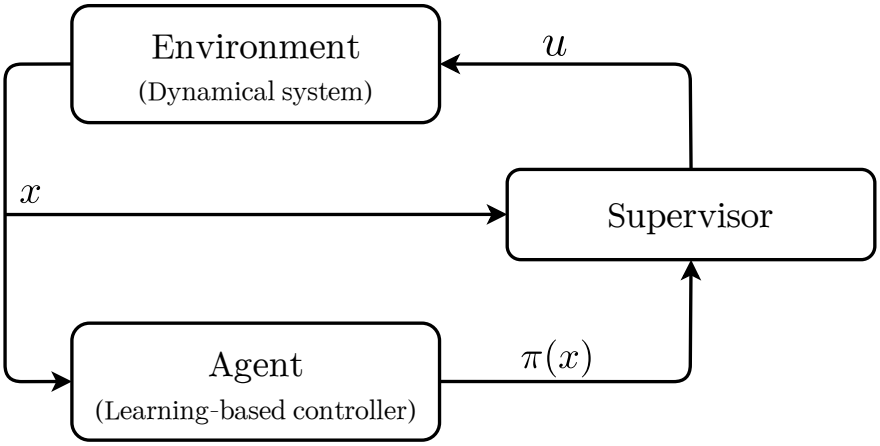


Figure 2.1.: SRL framework with the supervisor that decides the actual control action applied on the system.

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 $t = 0$.

Remark 1. *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. 2.1). 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 $\mathcal{S}$ depends on the corrective controller $K(x)$. A good choice of controller $K(x)$ provides a large enough safe region $\mathcal{S}$ such that the system maintains sufficient flexibility under the policy $\pi(x)$.

One essential part of the SRL framework is to represent the safe region as accurately as possible, however, the exact calculation of the ROA is usually not feasible [100]. In literature, a sum-of-squares (SOS) programming approach is widely used to get at least an inner approximation of the ROA [101] (see Appendix A for details). Since this approximation is 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 [102] which hinders its application on complex dynamical systems. Although there exist methods to relax the computational limitation of SOS programming [102], it is still difficult to effectively update the high-dimensional approximation of the ROA obtained through these methods. Thus, in next chapter we propose a practically implementable SRL framework for complex dynamical systems.

Remark 2. *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 [103], invariance functions [104] or barrier functions [105] can be used to construct a safe region. The SRL framework is generally compatible with these different concepts, as long as a closed and control invariant safe region can be defined.*

Safe Reinforcement Learning Based on Physically Inspired Model Order Reduction

3.

In this chapter, we introduce a general SRL framework that is designed for complex dynamical systems. According to the aforementioned supervisory control strategy, safety in learning is achieved by switching the actual applied controller between the reinforcement learning-based controller and a predefined corrective controller. Such a switching happens when the system is believed to be approaching the boundary of the safe region. In order to solve the challenging problem of dealing with complex dynamics, we employ a physically inspired MOR technique to construct a simplified system model that is able to provide estimates about the safe region. For having more accurate safety estimates, the safe region is updated online through an online adaptation method that utilizes the actual execution results of the corrective controller. Note that, due to the computational difficulty of calculating the exact safe region for complex dynamical systems [77], we have to relax the absolute safety guarantee by allowing a reasonable amount of failures. The proposed SRL framework can learn effectively from those failures, and later avoid similar dangerous behaviours.

The remainder of this chapter is organized as follows: an overview of the proposed SRL framework is first given in Section 3.1. Thereafter, the practical realization of the SRL framework on complex dynamical systems is proposed. In Section 3.2, we first explain how to initialize the SRL framework by using physically inspired MOR. Then, we propose an online adaptation method in Section 3.3 that modifies the identified safe region based on the observed original system's behaviour. An algorithm that summarizes the proposed SRL framework is presented in Section 3.4. In Section 3.5, three examples are given to demonstrate the performance of the proposed SRL framework; followed by a discussion of several aspects of the framework in Section 3.6 and a summary in Section 3.7.

3.1. Overview of the Approach

In this section, we present an overview of the proposed SRL framework that is implementable on complex dynamical systems. It uses the same supervisory control strategy as explained in Section 2.3, where the actual applied action is switched between the learning-based controller $\pi(x)$ and the corrective controller $K(x)$. The safe region $\mathcal{S}$ is defined by using the ROA of the corrective controller $K(x)$ such as given in Definition 1. However, in order to overcome the computational limitations in determining a safe region under highly nonlinear and high-dimensional system dynamics, we utilize a physically inspired MOR technique to extract a simplified system model from the original complex dynamical system.

As illustrated in Fig. 3.1, since no actual data is available prior to the learning process,

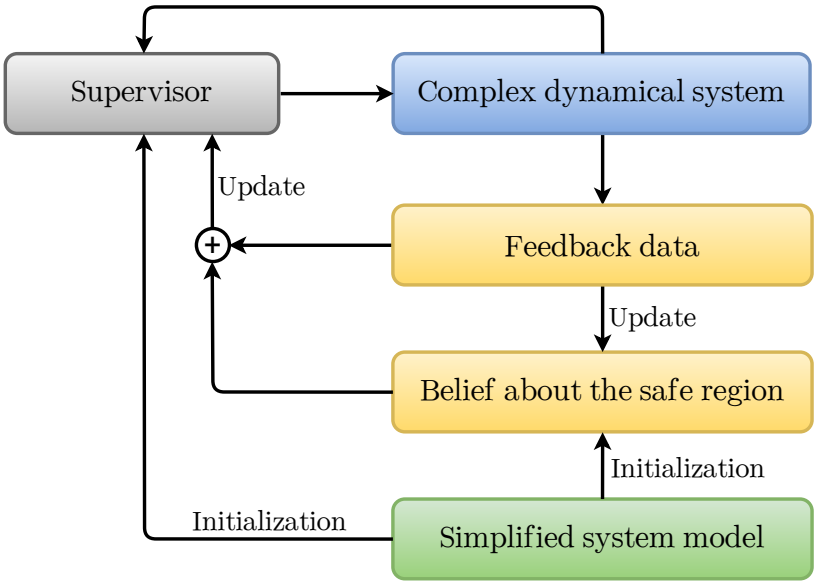


Figure 3.1.: Overview of the practical realization of the SRL framework on complex dynamical systems. The supervisor and the belief about the safe region is initialized by using a simplified system model and is updated through feedback data.

the simplified system model provides an initial belief about the safe region and is used to initialize the supervisor. To do this, the safe region of the simplified system is first computed through SOS programming or identified via examining the influence of important physical features. Then by learning a function approximator from the obtained safe region of the simplified system, the supervisor is constructed accordingly. Details about the initialization of the SRL framework are given in Section 3.2.

Due to the order reduction, it is unavoidable that the initial safety estimates might be inaccurate. Hence, for having reliable safety estimates that reflect the actual safe region of the original system, we propose an online adaptation method to update our belief about the safe region. The central idea is to use the actual execution results of the corrective controller, which are collected during the learning process and referred to as feedback data, as another belief source for estimating the safe region. The beliefs derived from the feedback data and the simplified system are later fused together for generating more accurate safety estimates. While the learning-based policy $\pi(x)$ is updated through reinforcement learning methods, a more reliable supervisor is learned simultaneously by using these better safety estimates. Details about the online adaptation method are given in Section 3.3.

3.2. Initialization of the Supervisor with Simplified System Model

In this section, we explain how to initialize the supervisor of the SRL framework by using a simplified system model that is identified via physically inspired MOR. A brief introduction to the property of the simplified system and its connection to the original system is first presented in Section 3.2.1. Then, we discuss reasons that cause inaccuracy in estimating the safety with the simplified system model and present a probabilistic form of safety estimate in

Section 3.2.2. In Section 3.2.3, we show how to initialize the supervisor by using a function approximator.

3.2.1. Simplified System

In practical engineering problems, various MOR techniques are used when the direct operation over the original system is computationally infeasible [41], e.g. physically inspired approaches [37], balanced truncation [38], system immersion [39] or abstraction [40]. In this chapter, we perform the MOR by using physically inspired approaches. In general, if the original system is represented by (2.19), then by considering important physical features, the MOR finds a simplified system

$$\dot{x}_s = f_s(x_s) + g_s(x_s)u_s, \tag{3.1}$$

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 state mapping of the MOR, 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 $\mathcal{S}_s$, can be estimated through SOS programming [101] or derived by using known information about system dynamics [81].

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

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

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 $\Phi(t; x)$ corresponds to a safe trajectory of the simplified system. Thus it motivates the initialization of the safe region of the original system $\mathcal{S}$ by using the safe region of the simplified system $\mathcal{S}_s$. In general, if the original system state x corresponds to a safe simplified system state $x_s \in \mathcal{S}_s$, then a suitable corrective controller $K(x)$ is most probably also able to control the original system state back to a safe state. This can, e.g., be achieved if there exists a corrective controller $K(x)$ that realizes ideal trajectory matching

$$\Psi(\Phi(t; x)) = \Phi_s(t; x_s), \tag{3.2}$$

where $\Phi_s(t; x_s)$ is the reference trajectory obtained from the simplified system that converges to the origin (see Fig. 3.2). See Remark 3 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 SRL framework.

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

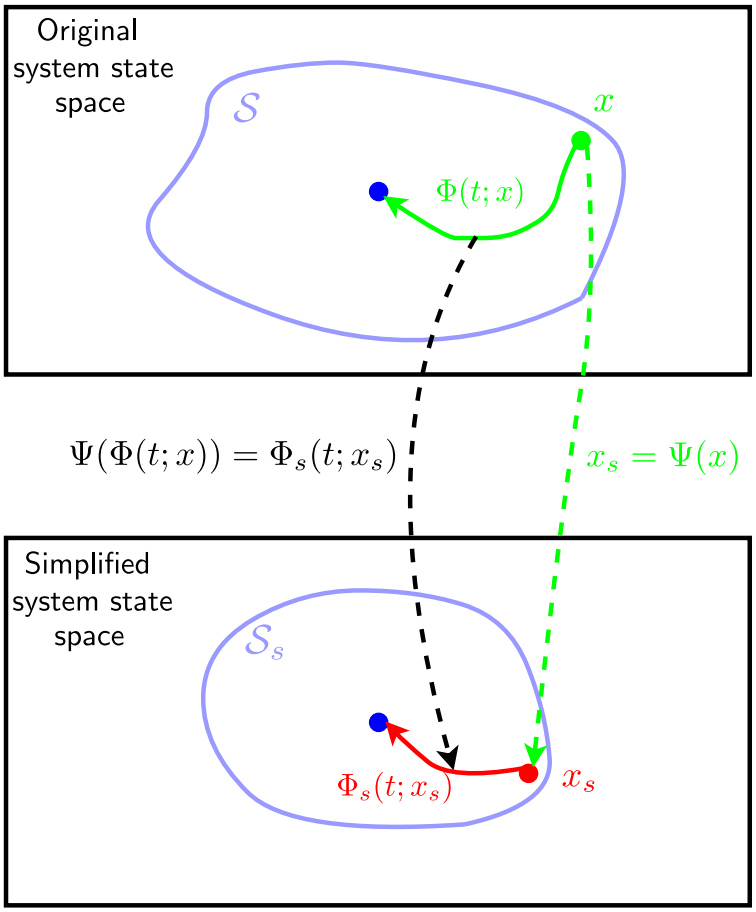


Figure 3.2.: For an original system state x and its corresponding simplified system state x_s, if $x_s \in \mathcal{S}_s$, then it is most likely that $x \in \mathcal{S}$ can be achieved by e.g. solving $\Psi(\Phi(t;x)) = \Phi_s(t;x_s)$.

Remark 3. *To realize the ideal trajectory mapping given as (3.2), one possible choice for the corrective controller $K(x)$ is minimizing the discrepancy between the mapped trajectory $\Psi(\Phi(t;x))$ and the reference trajectory $\Phi_s(t;x_s)$. Hence we have the following optimization problem*

$$\min_u ||\nabla_x \Psi(x)(f(x) + g(x)u) - \dot{x}_s||, \ \ s.t. \ \ u \in \mathcal{U}, \tag{3.3}$$

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 [104].

3.2.2. 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.

First, due to different control input spaces, the ideal trajectory matching represented by (3.2) may not be possible for all original system states x. Thus, the safety of an original

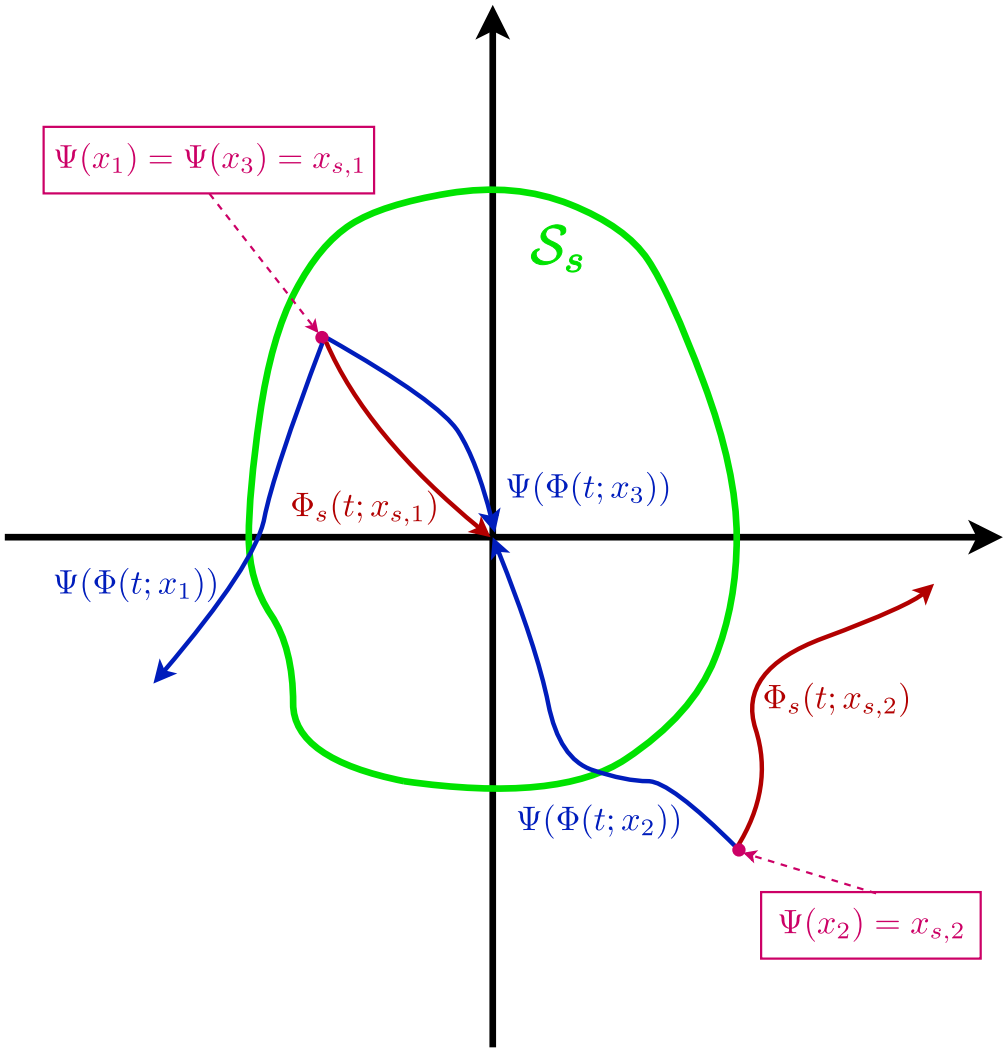


Figure 3.3.: Reasons for the inaccuracy of estimating the safety of an original system state through the simplified system. The original system state x does not always have the same safety property as its corresponding simplified system state x_s, and different original system states mapped to the same simplified system state may have different safety properties.

system state x is not guaranteed by the safety of its corresponding simplified system state x_s (e.g. $\Psi(\Phi(t; x_1))$ in Fig. 3.3). Second, 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 $\mathcal{S}_s$, the original system state x may still be a safe state (e.g. $\Psi(\Phi(t; x_2))$ in Fig. 3.3), 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. Third, 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. 3.3). 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.

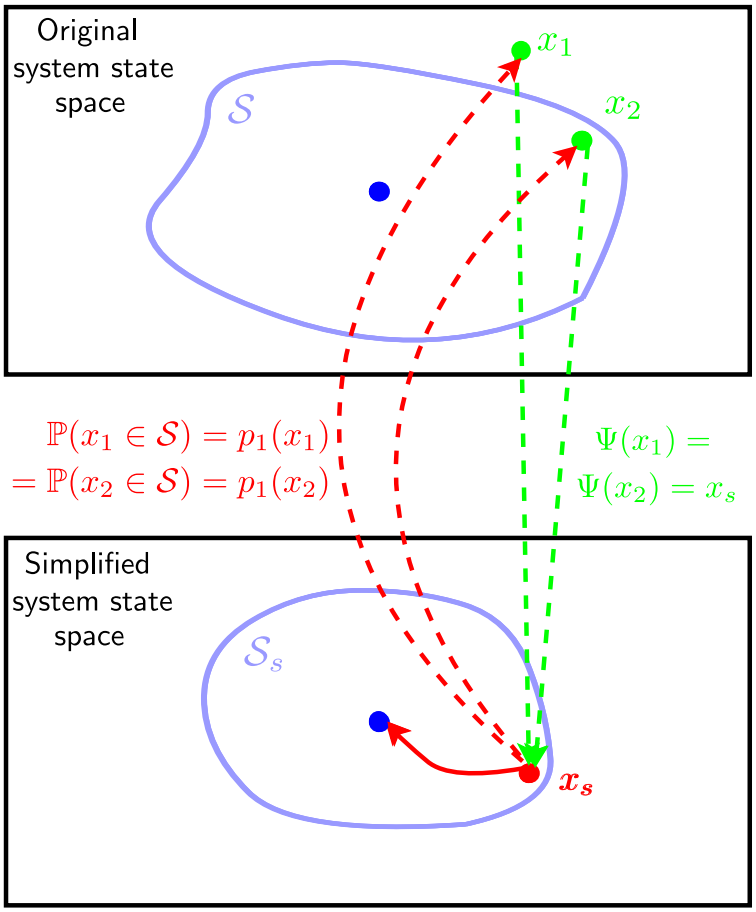


Figure 3.4.: Different original system states x mapped to the same simplified system state x_s have the same probabilistic estimate of safety.

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 way

$$\begin{aligned} \mathbb{P}(x \in \mathcal{S}) &= \underbrace{\mathbb{P}(x \in \mathcal{S} | \Psi(x) \in \mathcal{S}_s)}_{p_1(x)} \mathbb{P}(\Psi(x) \in \mathcal{S}_s) \\ &+ \underbrace{\mathbb{P}(x \in \mathcal{S} | \Psi(x) \notin \mathcal{S}_s)}_{p_2(x)} \mathbb{P}(\Psi(x) \notin \mathcal{S}_s), \end{aligned} \tag{3.4}$$

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 $\mathcal{S}_s$ is fixed, we have either $\mathbb{P}(\Psi(x) \in \mathcal{S}_s) = 1$ or $\mathbb{P}(\Psi(x) \notin \mathcal{S}_s) = 1$. Since $\mathbb{P}(\Psi(x) \in \mathcal{S}_s) + \mathbb{P}(\Psi(x) \notin \mathcal{S}_s) = 1$, (3.4) is simplified with the corresponding conditional probability as

$$\mathbb{P}(x \in \mathcal{S}) = \begin{cases} p_1(x), & \text{if } \Psi(x) \in \mathcal{S}_s, \\ p_2(x), & \text{if } \Psi(x) \notin \mathcal{S}_s, \end{cases} \tag{3.5}$$

which represents the probabilistic estimate of safety of an original system state x (see Fig. 3.4).

3.2.3. Supervisor Initialization

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

$$F(x) = \mathbb{P}(x \in \mathcal{S}) \sim [0, 1], \tag{3.6}$$

which gives the probability of a given original system state x being inside the safe region of the original system $\mathcal{S}$. Then, with a probability threshold p_t, the previous supervisor (2.22) is transformed into the following form

$$u = \begin{cases} \pi(x), & \text{if } t < t^{'}, \\ K(x), & \text{else,} \end{cases} \tag{3.7}$$

where $t^{'}$ denotes the first time point that the probability obtained from the function approximator is not larger than the threshold, i.e., $F(x) \leq p_t$. 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 [106].

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 (3.5), the probabilistic estimate of safety of each sample is calculated by using the safe region of the simplified system $\mathcal{S}_s$. We set the probability distributions to constants $p_1(x) = \overline{p}$, $p_2(x) = \underline{p}$, and obtain the initial estimate

$$\mathbb{P}_{\text{init}}(x_i \in \mathcal{S}) = \begin{cases} \overline{p}, & \text{if } \Psi(x_i) \in \mathcal{S}_s, \\ \underline{p}, & \text{if } \Psi(x_i) \notin \mathcal{S}_s, \end{cases} \tag{3.8}$$

where $0 < \underline{p} < p_t < \overline{p} < 1$. 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 3.3.3. An example is given in the following.

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

$$x_s = \Psi(x) = [x_e, x_f]^T = [x_a + x_b, x_c + x_d]^T, \tag{3.9}$$

$$\mathcal{S}_s = \{x_s \mid x_e^2 + x_f^2 \leq 9\}. \tag{3.10}$$

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

The function approximator $F(x)$ is then initialized by training with all sampled states and their corresponding estimates obtained from (3.8). In general, this initialization is expected to result in a restricted supervisor. To increase the performance of the SRL framework, we update the supervisor with an online adaptation method that is described in Section 3.3.

Remark 4. *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 chapter. 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 a 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.

3.3. Online Adaptation of the Safe Region

The simplified system model provides initial safety estimates and enables a proper initialization of the supervisor. After the initialization, the SRL framework can be applied to the actual complex dynamical system. 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 $\mathcal{S}$. 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 contains two elements $\{x, \mathbb{P}(x \in \mathcal{S}) = \{1, 0\}\}$. x is the original system state where the corrective controller $K(x)$ is activated. $\mathbb{P}(x \in \mathcal{S}) = \{1, 0\}$ 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 $\mathbb{P}(x \in \mathcal{S}) = 1$, and negative feedback data if $\mathbb{P}(x \in \mathcal{S}) = 0$. The set of all feedback data is denoted as $\mathcal{X}_{\text{real}}$.*

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 $\mathcal{S}$ is updated during the learning process such that a reliable supervisor is obtained through the online adaptation method.

In this section, we first introduce a mathematical tool called belief function theory that is able to consider uncertainty in the probabilistic estimate of safety. Then, two belief maps, one prior and one feedback, are constructed based on the belief function theory for representing the belief about the safe region. Later these two belief maps are fused together for deriving a combined belief map that gives more accurate safety estimates. The supervisor of the SRL framework is thus updated by using the feedback data as well as the probabilistic estimates of safety obtained from the combined belief map. Details of the proposed online adaptation method are given in the following subsections.

3.3.1. Belief Function Theory

The probabilistic estimate of safety obtained from (3.5) comes from the simplified system. However, as the exact mathematical relationship between the safe region of the original system $\mathcal{S}$ and the safe region of the simplified system $\mathcal{S}_s$ is unclear, such an estimate is in

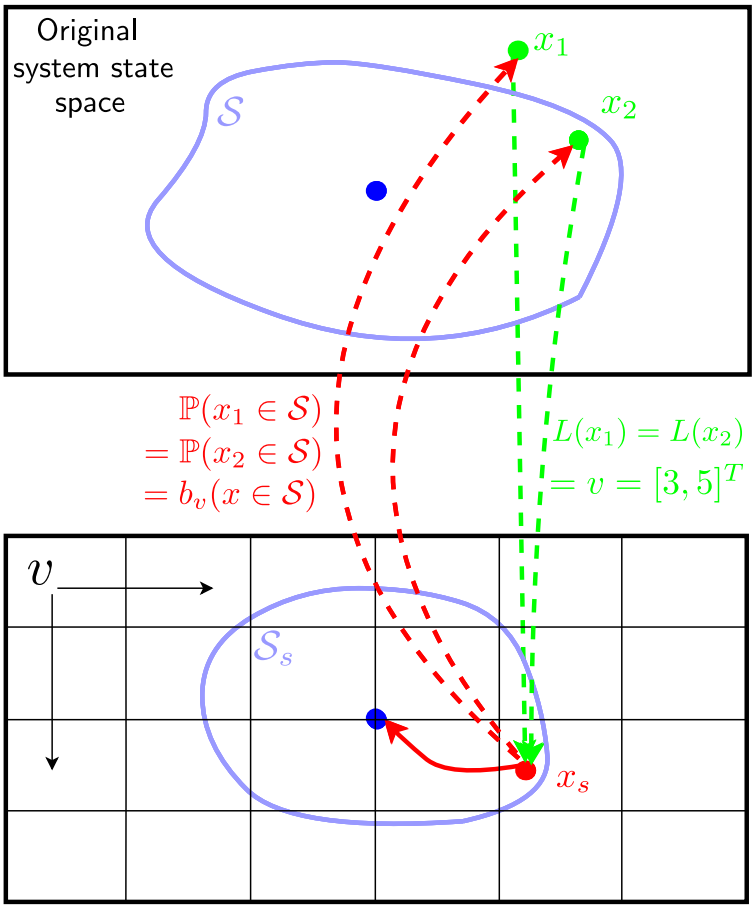


Figure 3.5.: The probabilistic estimate of safety is represented by the corresponding belief mass.

fact a subjective probability [107]. 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. $\mathbb{P}(\text{heads}) = \mathbb{P}(\text{tails}) = 0.5$. But with only one toss, the probability we obtain from the observation is either $\mathbb{P}(\text{heads}) = 1$ or $\mathbb{P}(\text{tails}) = 1$. 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 [108] into our SRL 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 [109]. 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.

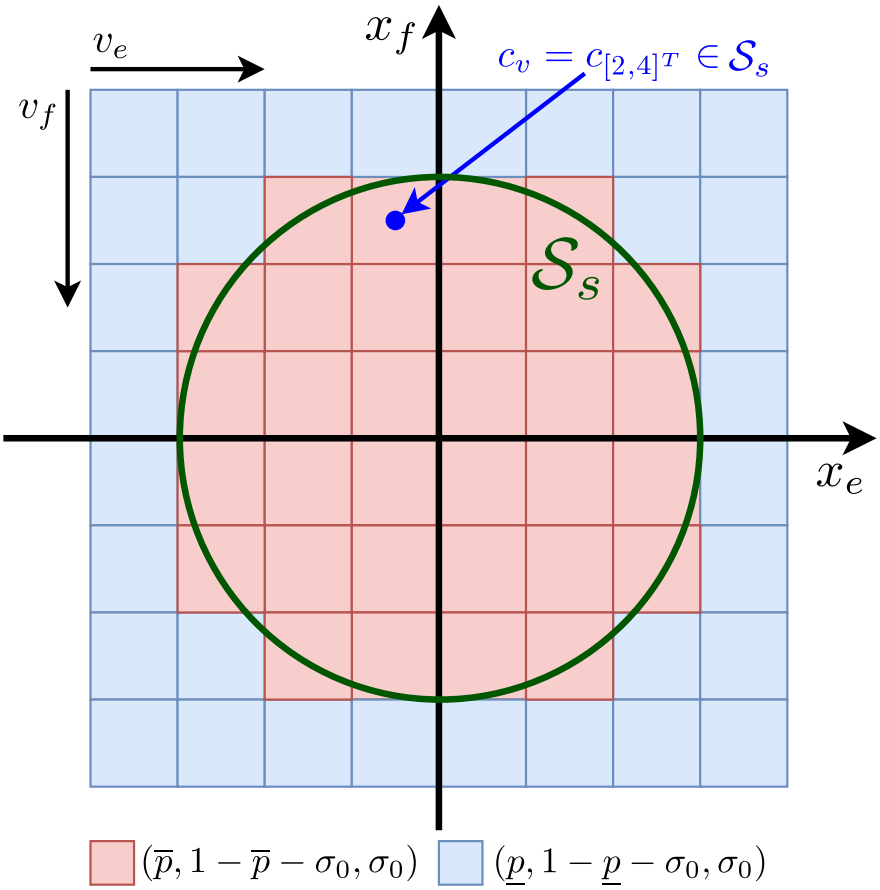


Figure 3.6.: The prior belief map $B^{\mathrm{P}}(v)$ of the Examples. The BBA for each index vector v is determined by the center point c_v of the corresponding grid cell.

3.3.2. 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 locating function $L(x)$.

Definition 3. *For an original system state x, the locating 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 locating 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 (3.5) to belief masses.

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

$$B(v) = (b_v(x \in \mathcal{S}), b_v(x \notin \mathcal{S}), \sigma_v), \tag{3.11}$$

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

$$b_v(x \in \mathcal{S}) + b_v(x \notin \mathcal{S}) + \sigma_v = 1, \tag{3.12}$$

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

The belief masses $b_v(x \in \mathcal{S})$ and $b_v(x \notin \mathcal{S})$ within each BBA represent the probabilities of the occurrence of two complementary events $x \in \mathcal{S}$ and $x \notin \mathcal{S}$, i.e. given the fact that the original system state x has the index vector v from the locating function $L(x)$, whether x is in the safe region of the original system $\mathcal{S}$ 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 $b_v(x \in \mathcal{S})$ as $b_{v,s}$ and $b_v(x \notin \mathcal{S})$ 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 (3.5) 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

$$\mathbb{P}(x \in \mathcal{S}) = b_v(x \in \mathcal{S})|_{v=L(x)} = b_{v,s}, \tag{3.13}$$

where the corresponding index vector $v = L(x)$ is determined by the locating function $L(x)$ (see Fig. 3.5). 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. 3.6). For original system states $x_1 = [0.1, 0.2, 0.3, 0.4]^T$, $x_2 = [0.2, 0.3, 0.4, 0.5]^T$, $x_3 = [0, 1.5, 1, 1.5]^T$, by locating the corresponding simplified system states $x_{s,1} = [0.3, 0.7]^T$, $x_{s,2} = [0.5, 0.9]^T$, $x_{s,3} = [1.5, 2.5]^T$ we thus have $L(x_1) = L(x_2) = [4, 5]^T$, $L(x_3) = [2, 6]^T$. The probabilistic estimates of safety are therefore given as $\mathbb{P}(x_1 \in \mathcal{S}) = \mathbb{P}(x_2 \in \mathcal{S}) = b_{[4,5]^T,s}$, $\mathbb{P}(x_3 \in \mathcal{S}) = b_{[2,6]^T,s}$.*

3.3.3. Prior and Feedback Belief Map

As aforementioned, the subjective uncertainties are caused by the unknown reliability of the simplified system and the insufficient amount of feedback 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 $B^{\mathrm{P}}(v)$ with the constants $\overline{p}$ and $\underline{p}$ from (3.8) as

$$\begin{aligned} B^{\mathrm{P}}(v) &= (b^{\mathrm{P}}_{v,s}, b^{\mathrm{P}}_{v,u}, \sigma^{\mathrm{P}}_v) \\ &= \begin{cases} (\overline{p}, 1 - \overline{p} - \sigma_0, \sigma_0), & \text{if } c_v \in \mathcal{S}_s, \\ (\underline{p}, 1 - \underline{p} - \sigma_0, \sigma_0), & \text{if } c_v \notin \mathcal{S}_s, \end{cases} \end{aligned} \tag{3.14}$$

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 $\mathcal{S}_s$ (e.g. $v = [2, 4]^T$ in Fig. 3.6). The subjective uncertainty σ^{P}_v reflects the reliability of the simplified system and is set to a constant σ_0 for all index vectors v.

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

$$\begin{cases} C_p^+(v) = C_p(v) + 1, & \text{if } \mathbb{P}(x \in \mathcal{S}) = 1, \\ C_n^+(v) = C_n(v) + 1, & \text{if } \mathbb{P}(x \in \mathcal{S}) = 0, \end{cases} \tag{3.15}$$

with the index vector $v = L(x)$ obtained from the locating 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

$$B^{\mathrm{F}}(v) = (b_{v,s}^{\mathrm{F}}, b_{v,u}^{\mathrm{F}}, \sigma_v^{\mathrm{F}}), \tag{3.16}$$

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

$$b_{v,s}^{\mathrm{F}} = \frac{C_p(v)}{C_p(v) + C_n(v)}(1 - \sigma_v^{\mathrm{F}}), \tag{3.17}$$

$$b_{v,u}^{\mathrm{F}} = \frac{C_n(v)}{C_p(v) + C_n(v)}(1 - \sigma_v^{\mathrm{F}}), \tag{3.18}$$

$$\sigma_v^{\mathrm{F}} = \alpha \cdot e^{-\beta \cdot (C_p(v) + C_n(v) - 1)}, \tag{3.19}$$

otherwise we set $B^{\mathrm{F}}(v)$ to an empty BBA. The subjective uncertainty σ_v^{F} depends on the amount of feedback data. If sufficient data is obtained, the subjective uncertainty σ_v^{F} approaches 0, and the belief mass $b_{v,s}^{\mathrm{F}}$ becomes the actual probability. Coefficients α and β 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 is 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 $B^{\mathrm{P}}(v)$ 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 $\mathcal{S}$, in the following we introduce a belief fusion operation to combine these two belief maps.

Remark 5. *The subjective uncertainty σ_v^{F} in (3.19) 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.*

3.3.4. Weighted Belief Fusion

Combining the prior belief map $B^{\mathrm{P}}(v)$ and the feedback belief map $B^{\mathrm{F}}(v)$ is referred to as weighted belief fusion [110], 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

$$\begin{aligned} B^{\mathrm{C}}(v) &= (b^{\mathrm{C}}_{v,s}, b^{\mathrm{C}}_{v,u}, \sigma^{\mathrm{C}}_{v}) \\ &= \begin{cases} B^{\mathrm{P}}(v), & \text{if } B^{\mathrm{F}}(v) \text{ is empty,} \\ B^{\mathrm{P}}(v) \oplus B^{\mathrm{F}}(v), & \text{if } B^{\mathrm{F}}(v) \text{ is not empty,} \end{cases} \end{aligned} \tag{3.20}$$

where the operator $\oplus$ is given as

$$b^{\mathrm{C}}_{v,s} = \frac{b^{\mathrm{P}}_{v,s}(1-\sigma^{\mathrm{P}}_{v})\sigma^{\mathrm{F}}_{v} + b^{\mathrm{F}}_{v,s}(1-\sigma^{\mathrm{F}}_{v})\sigma^{\mathrm{P}}_{v}}{\sigma^{\mathrm{P}}_{v} + \sigma^{\mathrm{F}}_{v} - 2\sigma^{\mathrm{P}}_{v}\sigma^{\mathrm{F}}_{v}}, \tag{3.21}$$

$$b^{\mathrm{C}}_{v,u} = \frac{b^{\mathrm{P}}_{v,u}(1-\sigma^{\mathrm{P}}_{v})\sigma^{\mathrm{F}}_{v} + b^{\mathrm{F}}_{v,u}(1-\sigma^{\mathrm{F}}_{v})\sigma^{\mathrm{P}}_{v}}{\sigma^{\mathrm{P}}_{v} + \sigma^{\mathrm{F}}_{v} - 2\sigma^{\mathrm{P}}_{v}\sigma^{\mathrm{F}}_{v}}, \tag{3.22}$$

$$\sigma^{\mathrm{C}}_{v} = \frac{(2-\sigma^{\mathrm{P}}_{v}-\sigma^{\mathrm{F}}_{v})\sigma^{\mathrm{P}}_{v}\sigma^{\mathrm{F}}_{v}}{\sigma^{\mathrm{P}}_{v} + \sigma^{\mathrm{F}}_{v} - 2\sigma^{\mathrm{P}}_{v}\sigma^{\mathrm{F}}_{v}}. \tag{3.23}$$

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 $B^{\mathrm{P}}(v)$ has no effect in making estimates.*

Proof. The following holds for the weighted belief fusion

$$\begin{aligned} \lim_{C_p(v)+C_n(v)\to\infty} b^{\mathrm{C}}_{v,s} &= b^{\mathrm{F}}_{v,s}, \\ \lim_{C_p(v)+C_n(v)\to\infty} b^{\mathrm{C}}_{v,u} &= b^{\mathrm{F}}_{v,u}, \\ \lim_{C_p(v)+C_n(v)\to\infty} \sigma^{\mathrm{C}}_{v} &= \sigma^{\mathrm{F}}_{v} = 0, \end{aligned} \tag{3.24}$$

which directly leads to Proposition 1. □

An example is given as follows.

Example 3. *Continued from previous examples, we now observe two feedback data $\{x_1, \mathbb{P}(x_1 \in \mathcal{S}) = 1\}$, $\{x_2, \mathbb{P}(x_2 \in \mathcal{S}) = 0\}$ 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*

$$B^{\mathrm{F}}([4, 5]^T) = (0.47, 0.47, 0.06), \tag{3.25}$$

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

$$B^{\mathrm{P}}([4, 5]^T) = (0.8, 0.1, 0.1). \tag{3.26}$$

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

$$B^{\mathrm{C}}([4, 5]^T) = (0.59, 0.34, 0.07). \tag{3.27}$$

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 $\mathbb{P}(x_4 \in \mathcal{S}) = b^{\mathrm{C}}_{[4,5]^T,s} = 0.59$.

In essence, the prior belief map $B^{\mathrm{P}}(v)$ serves as the initial belief that facilitates the SRL 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 $B^{\mathrm{C}}(v)$.

3.3.5. Supervisor Update

Although the set of feedback data $\mathcal{X}_{\mathrm{real}}$ contains the ground-truth information about the safe region of the original system $\mathcal{S}$, 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 3.2.3, 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 $\mathcal{X}_{\mathrm{real}}$, the samples have to be not in the neighborhood of any known states with a distance threshold d_t, i.e. $\forall x \in \mathcal{X}_{\mathrm{real}}$, it holds $||x - x_i|| > d_t$ with $i = 1 \ldots k_e$. Then according to (3.13), the probabilistic estimate of safety is assigned to each sample by using the combined belief map $B^{\mathrm{C}}(v)$ as

$$\mathbb{P}(x_i \in \mathcal{S}) = b^{\mathrm{C}}_{v_i,s}, \tag{3.28}$$

where the index vector $v_i = L(x_i)$ is computed by the locating function $L(x)$. This set of estimates is denoted as $\mathcal{X}_{\mathrm{est}}$ and is used to support the update of the supervisor. While the set of feedback data $\mathcal{X}_{\mathrm{real}}$ represents the absolute safety of already known states, $\mathcal{X}_{\mathrm{est}}$ predicts the safety of unknown states. In each supervisor update iteration, a new set of estimates $\mathcal{X}_{\mathrm{est}}$ 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 $\mathcal{X}_{\mathrm{real}}$ and the set of estimates $\mathcal{X}_{\mathrm{est}}$. In the learning process, we incrementally extend the set $\mathcal{X}_{\mathrm{real}}$ after new feedback data is obtained. The size k_e of the set of estimates $\mathcal{X}_{\mathrm{est}}$ is set to a fixed value since $\mathcal{X}_{\mathrm{est}}$ 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 $B^{\mathrm{P}}(v)$ 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 $B^{\mathrm{P}}(v)$ 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 SRL framework.

3.3.6. 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 [111].

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 (3.28). 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 probability threshold p_t in (3.7). Comparing with the true class label of feedback data, i.e. $\mathbb{P}(x \in \mathcal{S}) = \{1, 0\}$, 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 ACC = (TP + TN)/(TP + TN + TP + TN) and the false positive ratio FPR = FP/(FP + 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 $\mathcal{X}_{\mathrm{real}}$ 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 SRL 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 false positive 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 accuracy ACC and a small false positive ratio FPR. Otherwise the safe learning process is suggested to be terminated and restarted with another simplified system model.

Remark 6. *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.*

3.4. SRL Algorithm

The practical realization of the proposed SRL 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 reinforcement learning-based policy $\pi(x)$ is updated based on a predefined reward function, while the supervisor is updated according to the feedback data. Once new feedback data is obtained, the set of feedback data $\mathcal{X}_{\mathrm{real}}$ is extended, and the counters $C_p(v)$ and $C_n(v)$ are updated.

Algorithm 1 Practical realization of the proposed SRL framework.

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

Initialize $F(x)$ with $\mathcal{S}_s$ and k;
Initialize $B^{\mathrm{P}}(v)$ with $\mathcal{S}_s$;
Initialize $C_p(v)$ and $C_n(v)$ as zeros;
Set $\mathcal{X}_{\mathrm{real}}$ as an empty set;
$i = 0$, $j = 0$;
while $i < k_i$ **do**
    while current learning iteration is not terminated **do**
        if $F(x) > p_t$ **then**
            $u = \pi(x)$;
        else
            Iteration is terminated;
        end if
    end while
    Update $\pi(x)$;
    Apply $K(x)$;
    if $\forall x_i \in \mathcal{X}_{\mathrm{real}}$, $||x - x_i|| > d_t$ **or** $\mathbb{P}(x \in \mathcal{S}) = 0$ **then**
        Append $\mathcal{X}_{\mathrm{real}}$ with $\{x, \mathbb{P}(x \in \mathcal{S}) = \{0, 1\}\}$;
        Update $C_p(v)$ or $C_n(v)$;
        $j = j + 1$;
    end if
    if $j = k_u$ **then**
        Calculate $B^{\mathrm{F}}(v)$ from $C_p(v)$ and $C_n(v)$;
        Calculate $B^{\mathrm{C}}(v)$;
        Validation;
        if Valid **then**
            Create $\mathcal{X}_{\mathrm{est}}$ with $B^{\mathrm{C}}(v)$ and k_e, d_t;
            Update $F(x)$ with $\mathcal{X}_{\mathrm{real}}$ and $\mathcal{X}_{\mathrm{est}}$;
            $j = 0$;
        else
            Safe learning process is suggested to be terminated;
        end if
    end if
    $i = i + 1$;
end while

To prevent repetition, this modification only happens if all original system states stored in $\mathcal{X}_{\mathrm{real}}$ have distances to the new feedback data's state x larger than a threshold d_t, or the new feedback data represents a failure. Whenever k_u feedback data are obtained, the prior belief map $B^{\mathrm{P}}(v)$ and the feedback belief map $B^{\mathrm{F}}(v)$ are employed to form the combined belief map $B^{\mathrm{C}}(v)$. The set of estimates $\mathcal{X}_{\mathrm{est}}$ 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.

Parameters of the SRL framework are chosen by considering the following aspects. First, the probability threshold p_t decides on the aggressiveness of actions and is selected based on the actual task and the severity of failure. Second, the initialization of the prior belief map $B^{\mathrm{P}}(v)$ 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 p_t. Third, 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 α, β satisfy that $b^{\mathrm{C}}_{v,s} > p_t$ is obtained from (3.21) with $C_p(v) = 3$, $C_n(v) = 0$.

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

3.5. Experimental Results

In this section, three examples are presented. First, a simple two-link inverted pendulum example illustrates how the supervisor and the belief maps are updated based on the feedback data. Second, in a quadcopter control task the performance of the proposed SRL framework for complex dynamical systems is demonstrated both in simulations and in a real-world experiment. Third, a humanoid control experiment is provided to further show the applicability of the proposed SRL framework to various learning tasks and scenarios.

3.5.1. Two-link Inverted Pendulum

We first demonstrate the online adaptation of the supervisor for a two-link inverted pendulum given in Fig. 3.7. 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 MOR, a one-link inverted pendulum is constructed based on the CoM m_c and is considered as the simplified system (see Fig. 3.7). The simplified system state is $x_s = [\theta, \dot{\theta}]^T$ and the corresponding state mapping $x_s = \Psi(x)$ is

$$\theta = \arctan \frac{x_{m_c}}{h_{m_c}}, \quad \dot{\theta} = \frac{v_v}{l}, \tag{3.29}$$

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

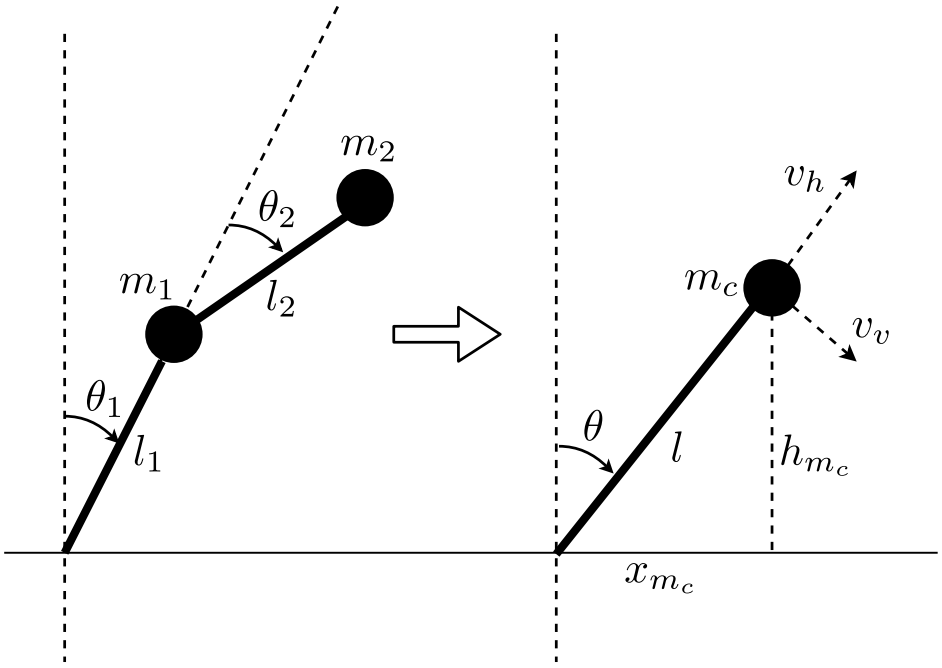


Figure 3.7.: Physically inspired MOR for a two-link inverted pendulum. By using the CoM, a one-link inverted pendulum approximates the two-link inverted pendulum dynamics.

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 state 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 $\mathcal{S}_s$ obtained from SOS programming provides an accurate estimate about the safe region of the original system $\mathcal{S}$. To demonstrate the online adaptation clearly, instead of using SOS programming, we select the safe region of the simplified system as $\mathcal{S}_s = \{x_s \mid \theta^2 + \dot{\theta}^2 \leq 0.6^2\}$, where the units of θ and $\dot{\theta}$ are rad and rad/s, 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 SRL framework used in this example are given in Table 3.1. 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 $\theta \in [-1.6 \text{ rad}, 1.6 \text{ rad}]$, $\dot{\theta} \in [-2 \text{ rad/s}, 2 \text{ rad/s}]$, and is discretized with 0.1 rad for θ and 0.1 rad/s for $\dot{\theta}$.

Fig. 3.8 gives the probability $\mathbb{P}(x \in \mathcal{S})$ from the function approximator $F(x)$ for the

Table 3.1.: Parameters of the SRL framework

	Inverted Pendulum	Crazyflie Simulation	Crazyflie Real-world	Humanoid
p_t	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
α	0.2	0.2	0.1	0.3
β	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_t	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

slice $\dot{\theta}_1 = \dot{\theta}_2 = 0$ in different supervisor update iterations, while Fig. 3.9 shows the slice $\theta_2 = \dot{\theta}_2 = 0$. The ground-truth values of safety $\mathbb{P}(x \in \mathcal{S}) = \{0, 1\}$ are presented for comparison. Fig. 3.10 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 $B^{\mathrm{C}}(v)$ equals to the prior belief map $B^{\mathrm{P}}(v)$. Due to the inaccuracy of the prior belief map $B^{\mathrm{P}}(v)$, the prediction differs from the ground-truth value (see Fig. 3.8a, Fig. 3.9a and Fig. 3.10a). However, after 10 updates of the supervisor, the accuracy of the prediction increases quickly. The combined belief map $B^{\mathrm{C}}(v)$ also changes iteratively according to the feedback data (see Fig. 3.8b, Fig. 3.9b and Fig. 3.10b). With more iterations, e.g. 30 iterations in Fig. 3.8c, Fig. 3.9c and Fig. 3.10c, 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 [81]. 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. 3.8d, Fig. 3.9d and Fig. 3.10d).

The accuracy ACC and the false positive ratio FPR of the combined belief map $B^{\mathrm{C}}(v)$ in different update iterations are given in Fig. 3.11. The probability 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 α and β for the feedback belief map $B^{\mathrm{F}}(v)$, several iterations are required until a wrong estimate obtained from the prior belief map $B^{\mathrm{P}}(v)$ is corrected by the feedback data. As a result, the belief about the safe region of the original system $\mathcal{S}$ 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. 3.3.

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 the supervisor. However in practice, we are only interested in locating the decision boundary,

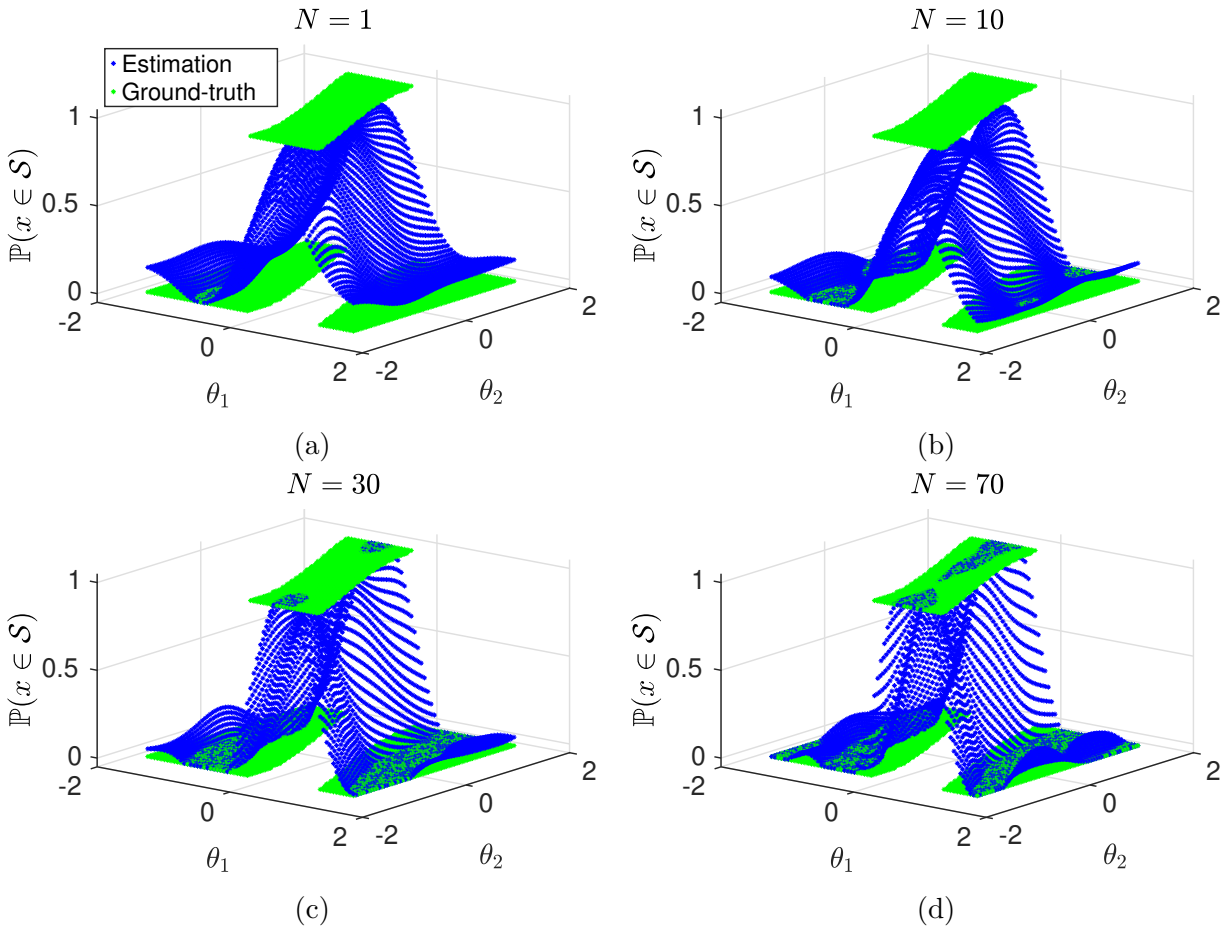


Figure 3.8.: Probability $\mathbb{P}(x \in \mathcal{S})$ from the function approximator $F(x)$ for the slice $\dot{\theta}_1 = \dot{\theta}_2 = 0$ in different supervisor update iterations $N = 1$, $N = 10$, $N = 30$ and $N = 70$. The ground-truth values of $\mathbb{P}(x \in \mathcal{S})$, obtained by testing the given corrective controller $K(x)$ at each sampled point, are presented for comparison.

as in the example we described in the next subsection involving a complex dynamical system.

3.5.2. Quadcopter Flight Control

To demonstrate the utility and the performance of the proposed SRL framework for complex and real-world dynamical systems, we control the Crazyflie[1], a lightweight nano quadcopter (see Fig. 3.12). The reinforcement 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) [112] from OpenAI Baselines[2]. The corrective controller is a PID controller which keeps the quadcopter hovering at a given height. By using the proposed SRL 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, $v_b = [v_x, v_y, v_z]^T$

[1] https://www.bitcraze.io/crazyflie-2/
[2] https://github.com/openai/baselines

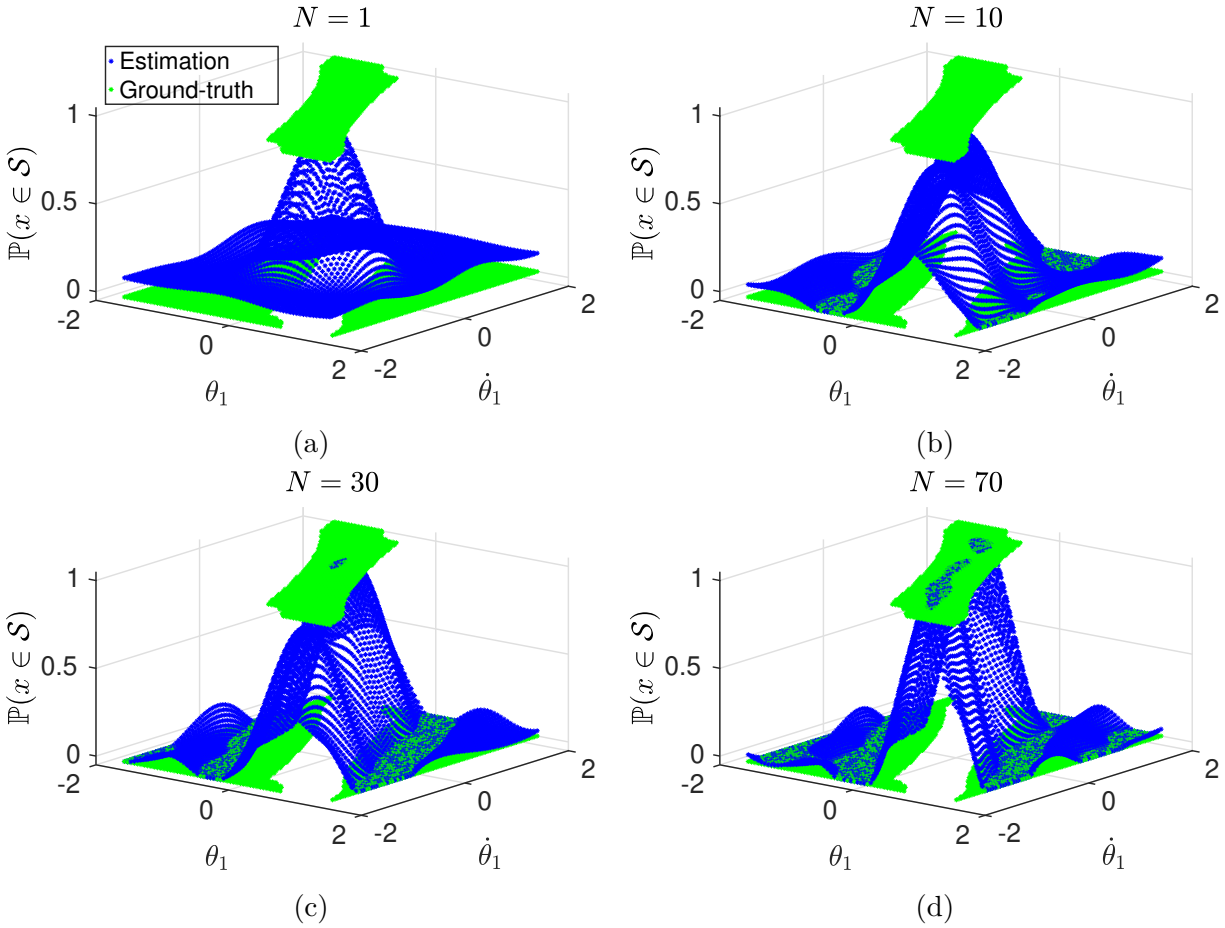


Figure 3.9.: Probability $\mathbb{P}(x \in \mathcal{S})$ from the function approximator $F(x)$ for the slice $\theta_2 = \dot{\theta}_2 = 0$ in different supervisor update iterations $N = 1$, $N = 10$, $N = 30$ and $N = 70$.

and $\omega_b = [\omega_r, \omega_p, \omega_y]^T$ are the linear and angular velocities in the body frame. 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 10 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, θ_p and their angular velocities ω_r, ω_p in roll and pitch directions. 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, 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 $\mathcal{S}_s$ is chosen as $\mathcal{S}_s = \{x_s \mid -4 \leq \omega_r \leq 4, -4 \leq \omega_p \leq 4\}$. 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 the next subsection.

There are four steps in each learning trial. First, the quadcopter takes off from the

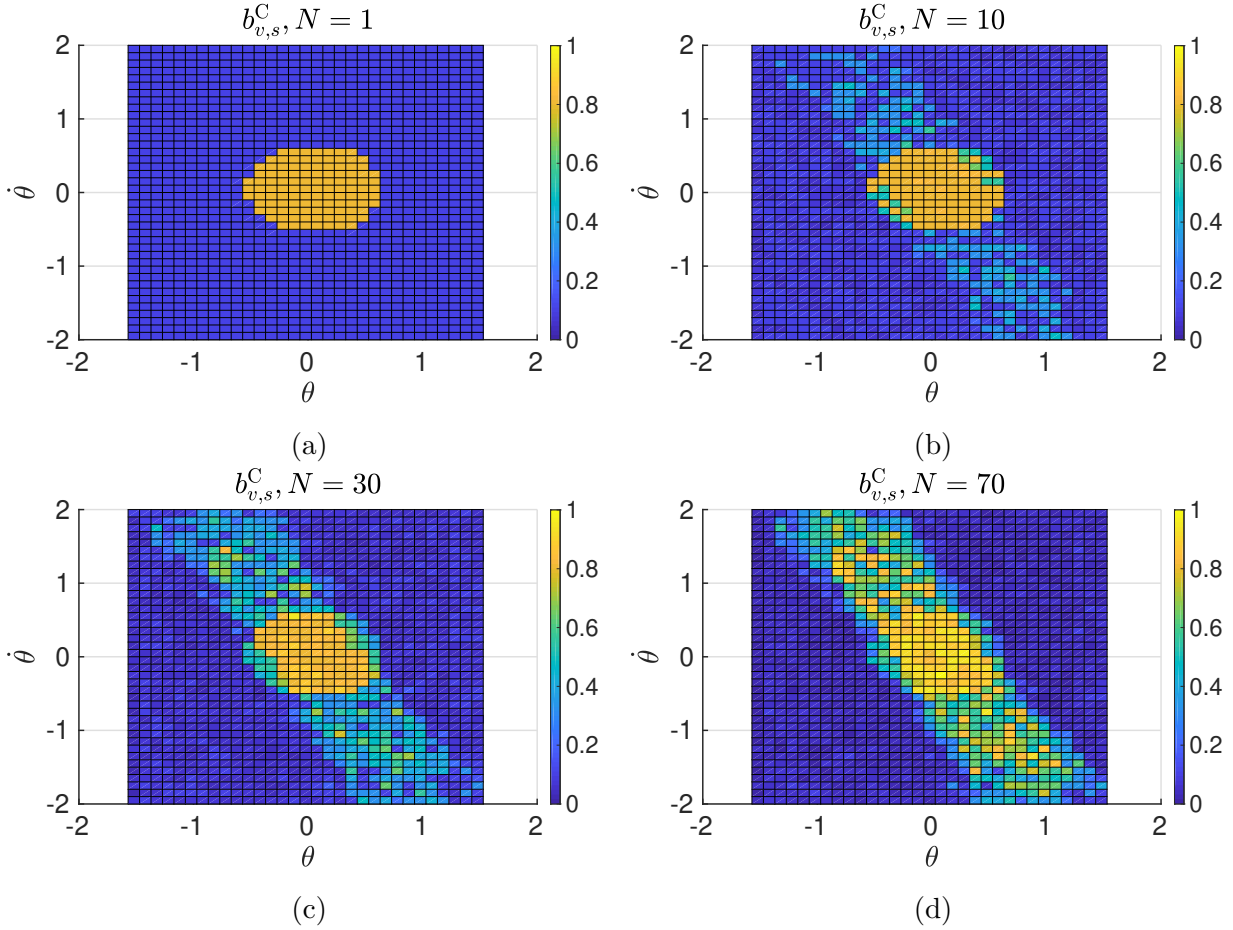


Figure 3.10.: The belief mass $b^{\mathrm{C}}_{v,s}$ from the combined belief map $B^{\mathrm{C}}(v)$ for different index vectors v in different supervisor update iterations $N = 1$, $N = 10$, $N = 30$ and $N = 70$.

ground and hovers at the position $p_g = [0, 0, 1\ \mathrm{m}]^T$. Second, the reinforcement learning-based controller $\pi(x)$ starts to control the quadcopter. Third, 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 $K(x)$ 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. $p_z < 0.7$ m. The corrective controller attempts to balance the quadcopter back to a safe hovering state. 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 given in [89].

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

Simulation

The learning process is simulated in Gazebo[3] with the Crazyflie model provided by [113] (see Fig. 3.13). The communication between the Gazebo simulator and the PPO algorithm

[3] `http://gazebosim.org/`

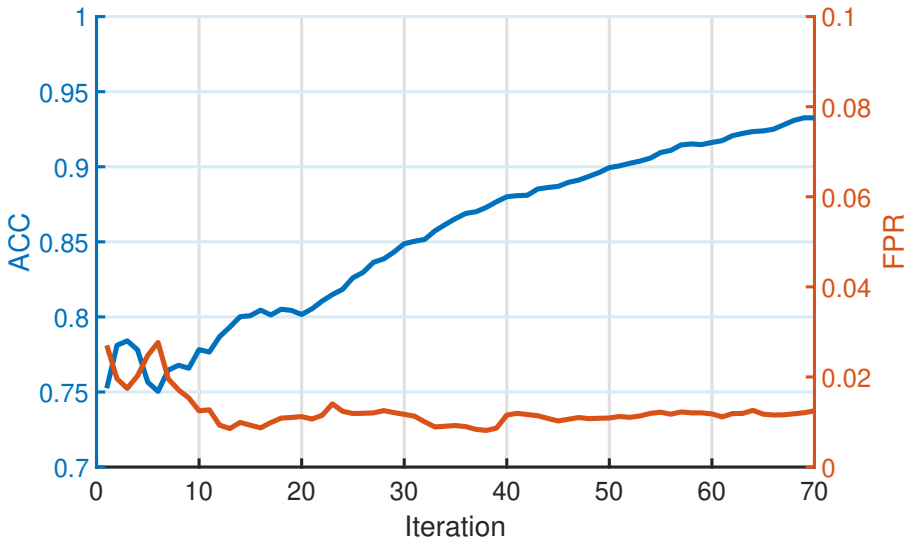


Figure 3.11.: Accuracy ACC and false positive ratio FPR of the two-link inverted pendulum example.

Table 3.2.: Hyperparameters of the PPO algorithm

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

is established through the Robot Operating System[4] using the Gym-Gazebo package [114].

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

- No supervisor (NS): no supervisor is implemented and the corrective controller is activated only when $p_z < 0.7$ m.
- 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 $F(x) < p_t$ with $p_t = 0.5$ in (3.7).
- With supervisor (WS): the proposed framework is implemented and three different probability thresholds are investigated: $p_t = 0.2$ (WS-0.2), $p_t = 0.5$ (WS-0.5) and $p_t = 0.8$ (WS-0.8).

The learning-based controller $\pi(x)$ controls the four motor speeds and runs at 200 Hz. The parameters of the SRL framework and the hyperparameters of the PPO algorithm used

[4] `https://www.ros.org/`

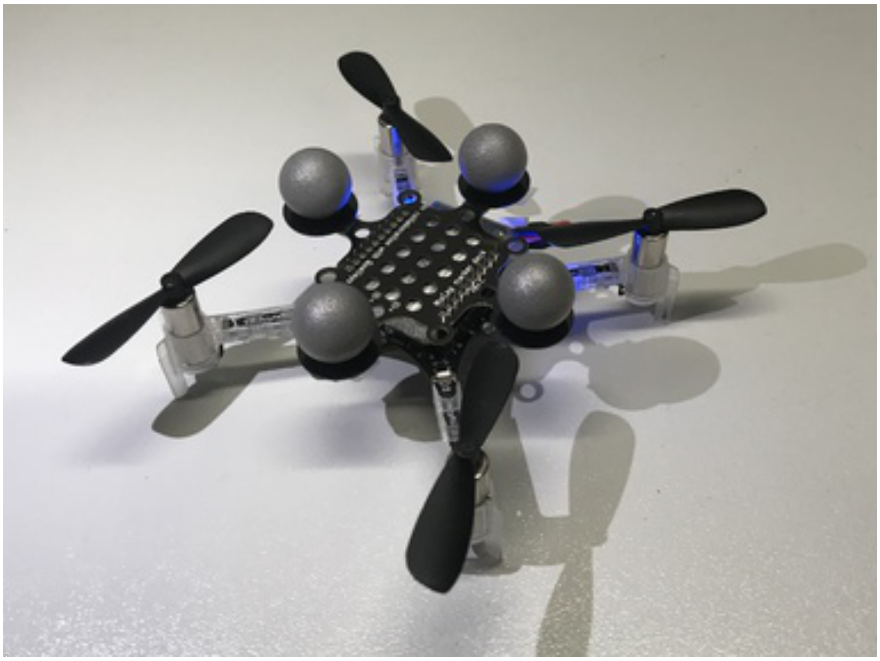


Figure 3.12.: Crazyflie with four tracking markers for the Qualisys motion capture system.

in this example are given in Table 3.1 and Table 3.2, respectively. The reward function is designed as

$$\begin{aligned} r(t) &= 1e5||p_{\text{goal}} - p_g(t-1)|| - 1e5||p_{\text{goal}} - p_g(t)|| \\ &\quad -||v_b(t) - v^d||^2 - 0.1||\omega_b(t)||^2 \end{aligned} \tag{3.30}$$

where $p_{\text{goal}} = 100v^d$ 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 this example. Starting with a random initial policy, we run the PPO algorithm for 307200 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. 3.14. 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 probability threshold p_t. While with a low threshold (WS-0.2) the supervisor has a minor effect on the learning process, 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. 3.15. Since a small threshold allows more risky actions, the probability threshold p_t 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 process of the combined belief map $B^{\text{C}}(v)$, where the initialization and the results in different iterations are given in Fig. 3.16 and Fig. 3.17, respectively. Apparently, a higher probability threshold p_t leads to a more restricted expansion process. If only feedback data is used (FO), the supervisor is initialized with the same prior belief map, as no feedback data is available prior to the learning process. After the first supervisor update iteration,

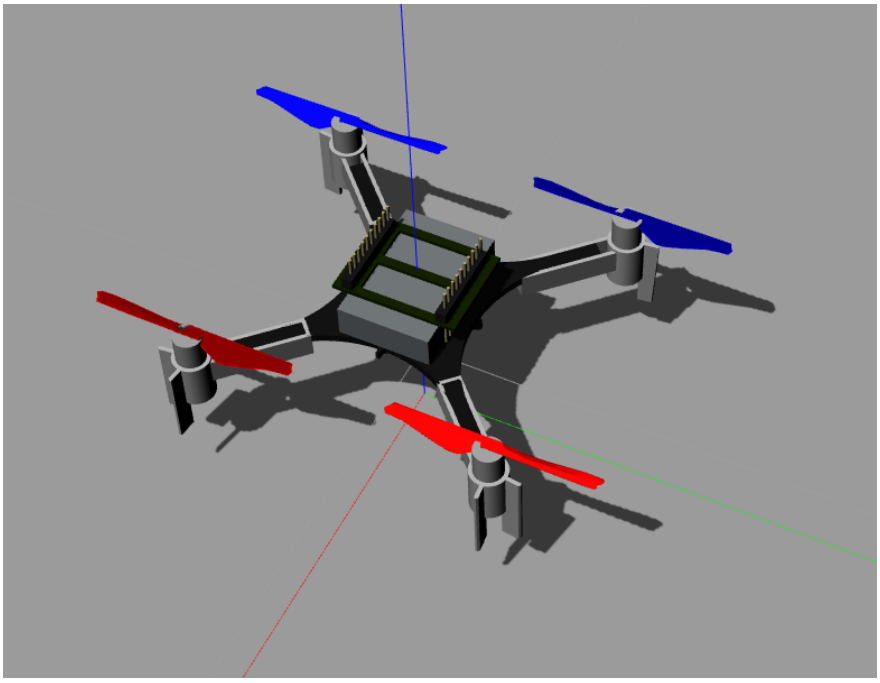


Figure 3.13.: Crazyflie model in Gazebo simulation environment.

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), 91% (WS-0.8). Note that, by only using the angular velocities certain failures cannot be separated, e.g. in Fig. 3.17 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.

Real-world experiment

The proposed SRL framework is also tested for a real Crazyflie (see Fig. 3.12). 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[5] motion capture system.

Considering the safety of the hardware, the learning process is performed only with supervisor and a moderate probability threshold $p_t = 0.5$ (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 $-0.5\ \mathrm{m} \leq p_x \leq 2\ \mathrm{m}$, $-0.4\ \mathrm{m} \leq p_y \leq 0.4\ \mathrm{m}$. 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 $\theta^d_g = [\theta^d_r, \theta^d_p, \theta^d_y]^T$

[5] `https://www.qualisys.com/`

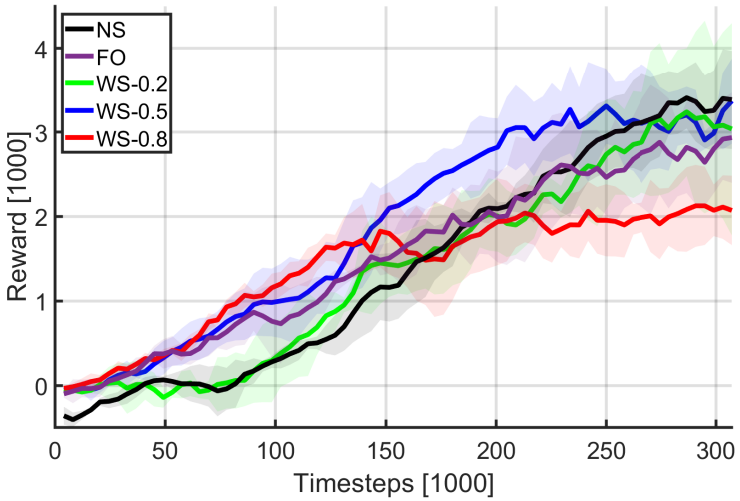


Figure 3.14.: Learning rewards of different learning conditions in simulation.

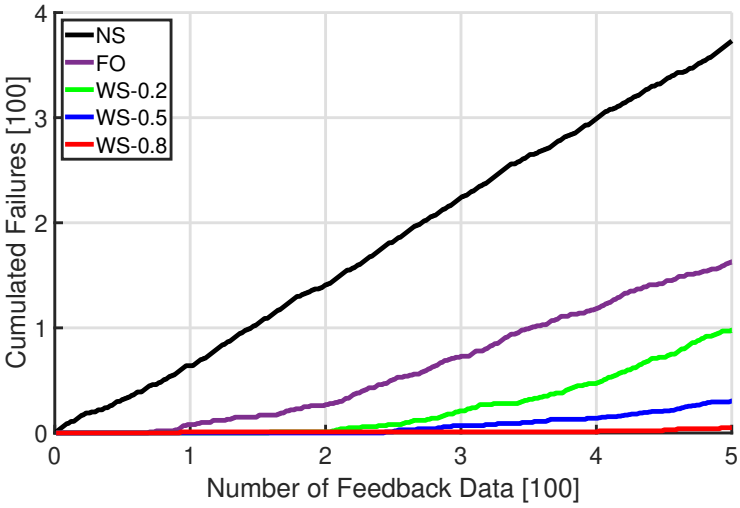


Figure 3.15.: Cumulated failures of different learning conditions in first 500 feedback data.

and the desired thrust t^d, i.e. $u = [\theta_g^d, t^d]$. 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 Table 3.1 and Table 3.2.

Starting with a random initial policy, we perform 30 PPO update iterations (1024 timesteps per each update) of the reinforcement learning-based controller. Details of the experimental results are presented in the supplementary video given in [89]. The reward and the accumulated failures are given in Fig. 3.18 and Fig. 3.19, 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 $b_{v,s}^{\mathrm{C}}$ of the combined belief map $B^{\mathrm{C}}(v)$ in different supervisor update iterations are shown in Fig. 3.20. In the early learning phase, due to the conservative estimate of the safe region, the reinforcement learning-based controller is quickly replaced

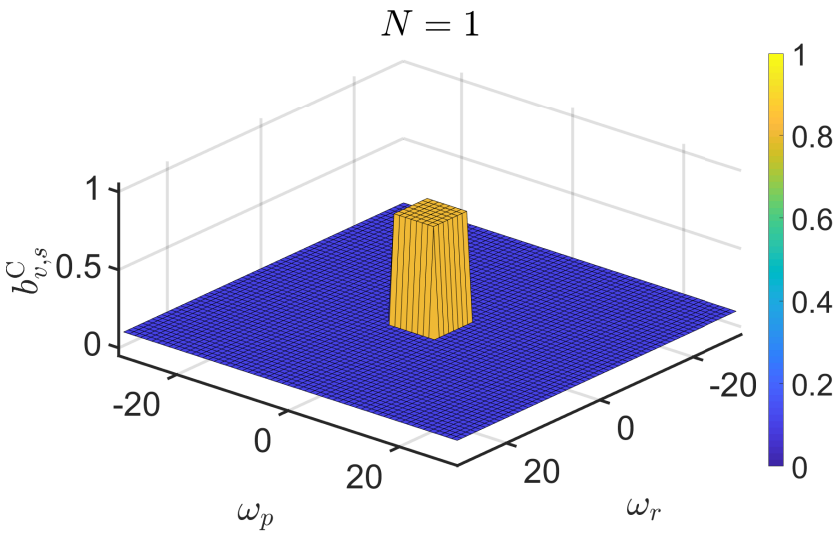


Figure 3.16.: Initialization ($N = 1$) of the belief mass $b^{\mathrm{C}}_{v,s}$ of the combined belief map $B^{\mathrm{C}}(v)$.

by the corrective controller. When more feedback data is 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 accuracy ACC and the false positive ratio FPR are given in Fig. 3.21. Since in the early learning phase the corrective controller is activated within the initial safe region, the accuracy 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 $B^{\mathrm{P}}(v)$. Once a more reliable estimate of the safe region is obtained, the accuracy remains high. However, we observe a high false positive ratio in this example. As the simplified system state consists only of angular velocities ω_r, 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, θ_p.

3.5.3. Humanoid Control

In this example, a humanoid robot is used to further demonstrate the applicability and the performance of the proposed SRL framework. The Atlas humanoid model (version 1) from DRCSIM[6] is utilized and is constrained to be able to move in the X-Z plane (see Fig. 3.22). 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 6 motor torque commands. A reinforcement 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 SRL 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.

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, $v_g =$

[6] `https://bitbucket.org/osrf/drcsim/`

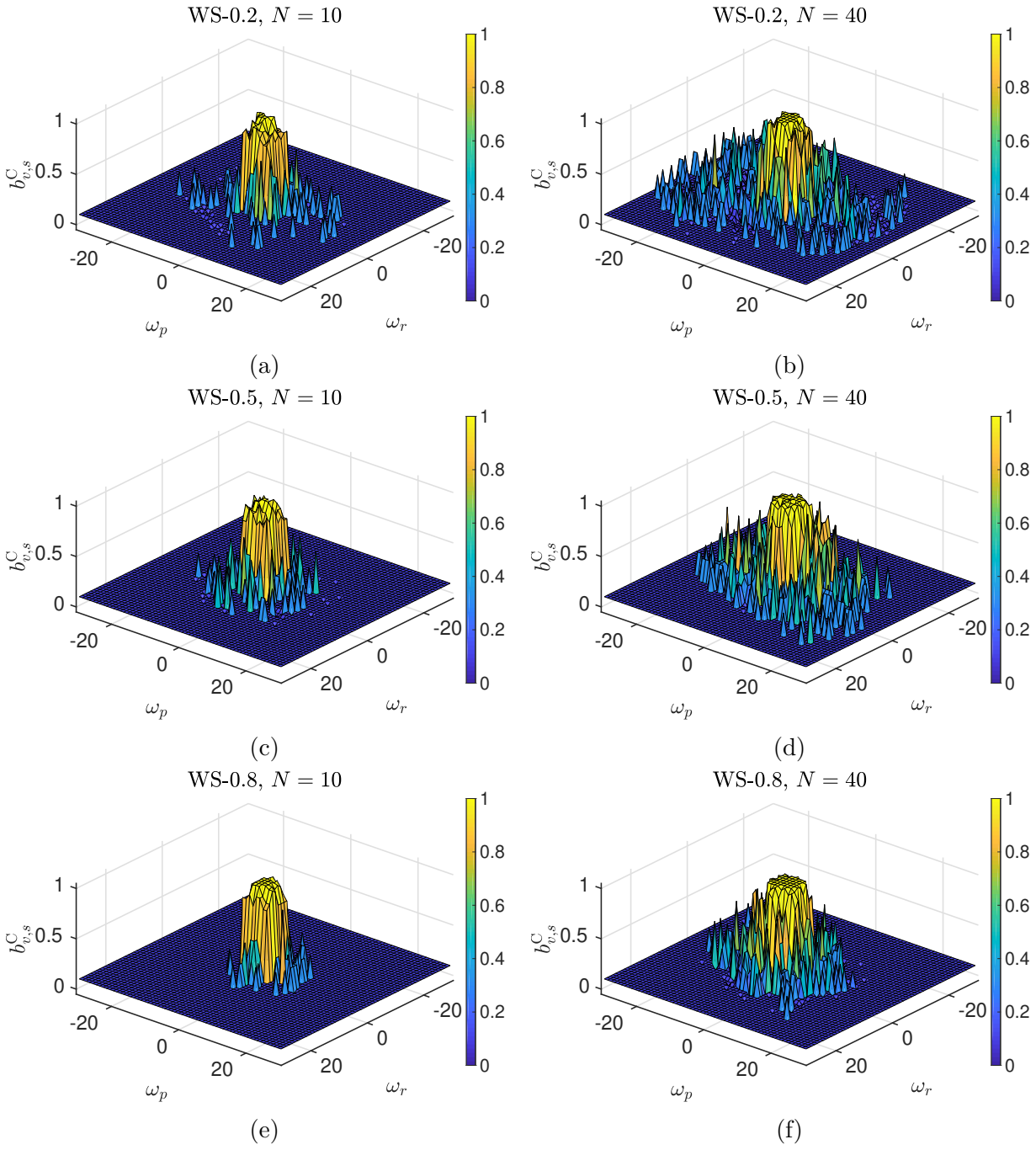


Figure 3.17.: The belief mass $b^{\mathrm{C}}_{v,s}$ of the combined belief map $B^{\mathrm{C}}(v)$ for conditions WS-0.2, WS-0.5 and WS-0.8 in the supervisor update iteration $N = 10$ and $N = 40$, respectively.

$[\dot{x}_g, \dot{z}_g, \dot{\theta}_g]^T$ are the body velocities. q and $\dot{q}$ are vectors of 6 joint angles and 6 joint velocities, respectively. As x_g, 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 dimensional. Based on the CoM m_c of the humanoid, an inverted pendulum is used as the simplified system (see Fig. 3.23). 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, while $\dot{x}_c$ and $\dot{z}_c$ are the velocities of

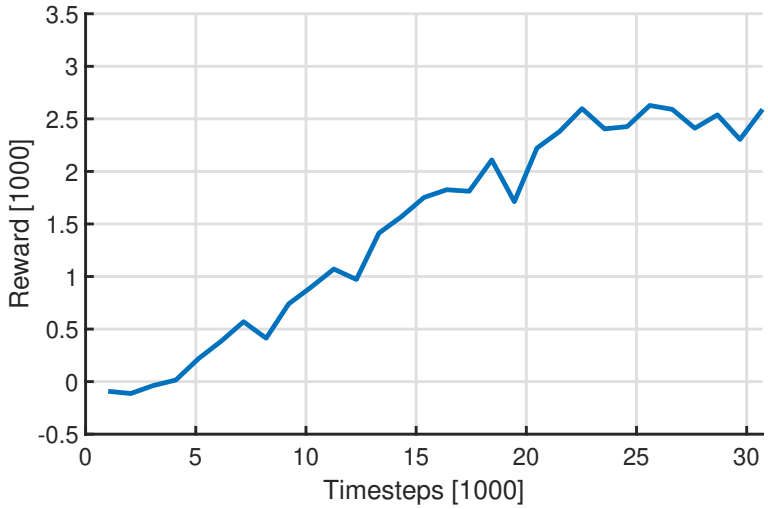


Figure 3.18.: Learning reward of the real-world quadcopter experiment.

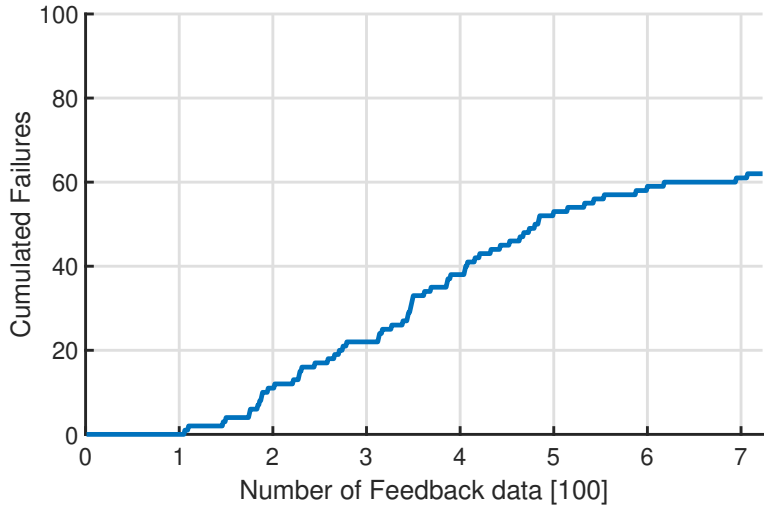


Figure 3.19.: Cumulated failures of the real-world quadcopter experiment.

the CoM [37]. Considering the physical limits of the system, the CoM properties are assumed to be within the range as $x_c \in [-1\text{ m}, 1\text{ m}]$, $z_c \in [0.4\text{ m}, 1\text{ m}]$, $\dot{x}_c, \dot{z}_c \in [-5\text{ m/s}, 5\text{ m/s}]$. The simplified system state space is discretized with 0.2 m for x_c, z_c and 1 m/s for $\dot{x}_c$, $\dot{z}_c$.

The corrective controller $K(x)$ implemented here is a 1-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 [115]. 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 SRL framework is replaced by a concept called N-step viable-capture basin [115]. 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

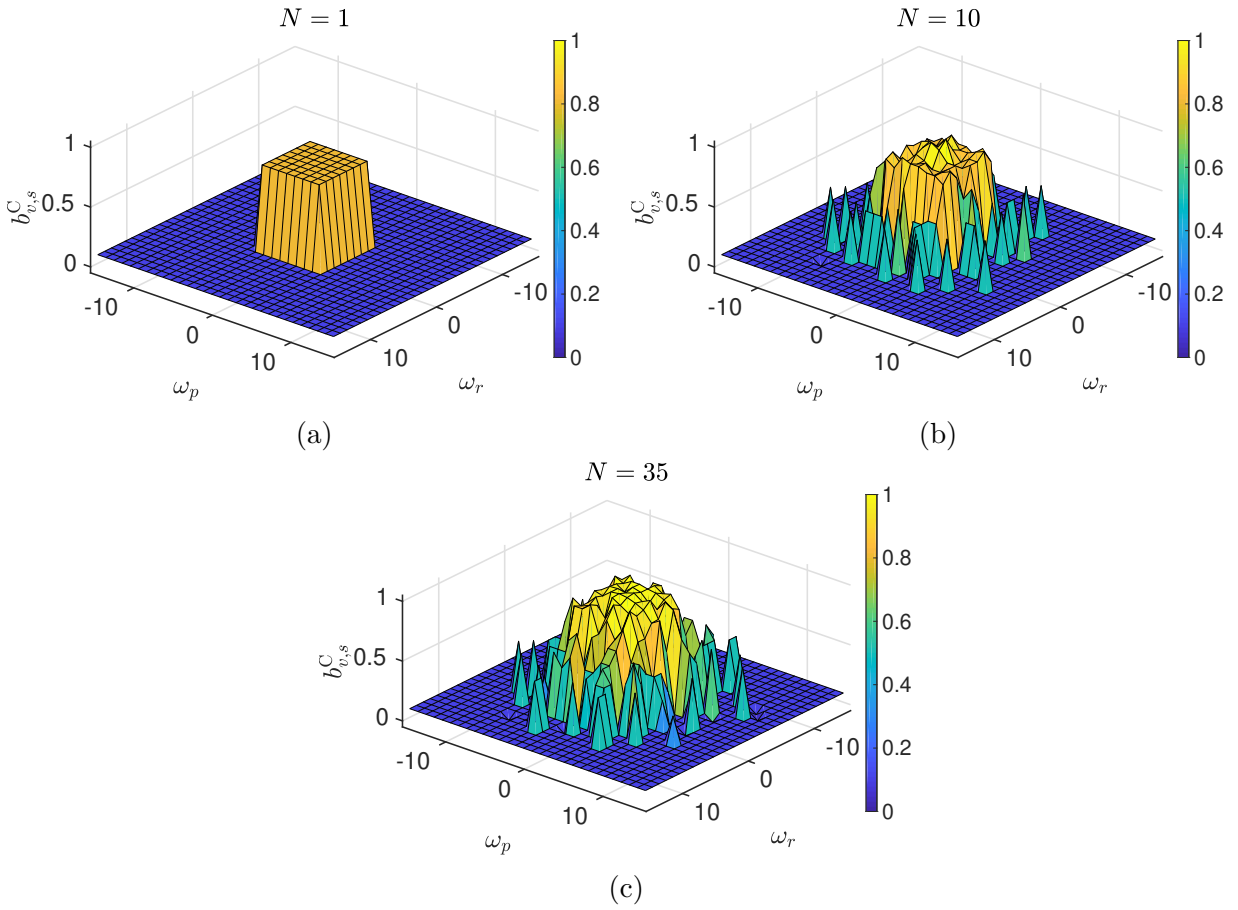


Figure 3.20.: The belief mass $b^{\mathrm{C}}_{v,s}$ of the combined belief map $B^{\mathrm{C}}(v)$ for different index vectors v in supervisor update iterations $N = 1$, $N = 10$, $N = 35$, respectively.

itself without stepping). In this example, we use the 1-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 1-step viable capture basin of the inverted pendulum, i.e. the safe region of the simplified system $\mathcal{S}_s$, is obtained by applying the approach described in [81].

The parameters used in this example are given in Table 3.1 and Table 3.2. The reward function is designed as

$$r(t) = 1e4(x_g(t) - x_g(t-1)) - 10{\theta_g}^2 - 2n_j \tag{3.31}$$

where n_j is the number of joints that are on the limit. 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 given in [89]. The overall success rate of the corrective controller is 79%. Note that, while the safety is increased with the proposed SRL framework, the learning performance is limited by the implemented 1-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

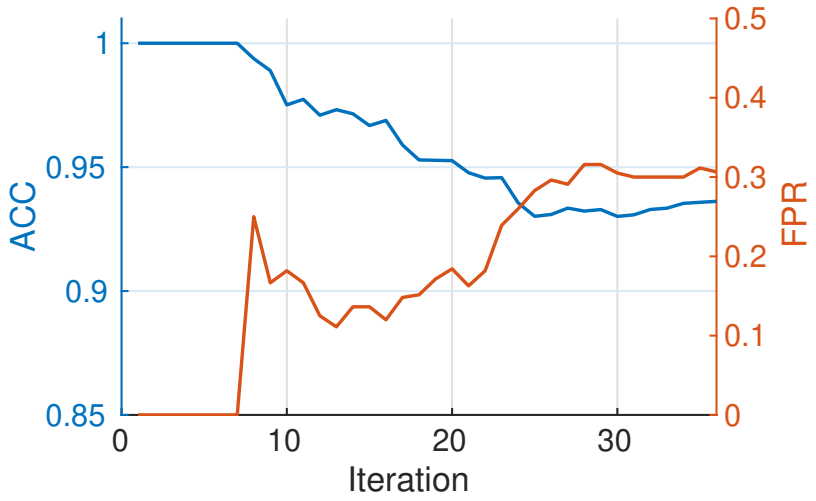


Figure 3.21.: Accuracy ACC and false positive ratio FPR of the combined belief map $B^{\mathrm{C}}(v)$ for the real-world quadcopter experiment.

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.

3.6. Discussion

In this chapter, we propose a SRL framework to increase the safety of reinforcement learning methods for autonomous systems. A supervisor is constructed to guide the exploration process and 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 SRL framework. Several critical features of the framework are discussed in this section.

3.6.1. Safety in Complex Dynamical Systems

In recent studies, e.g., [71], [75], 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 GPR model 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.

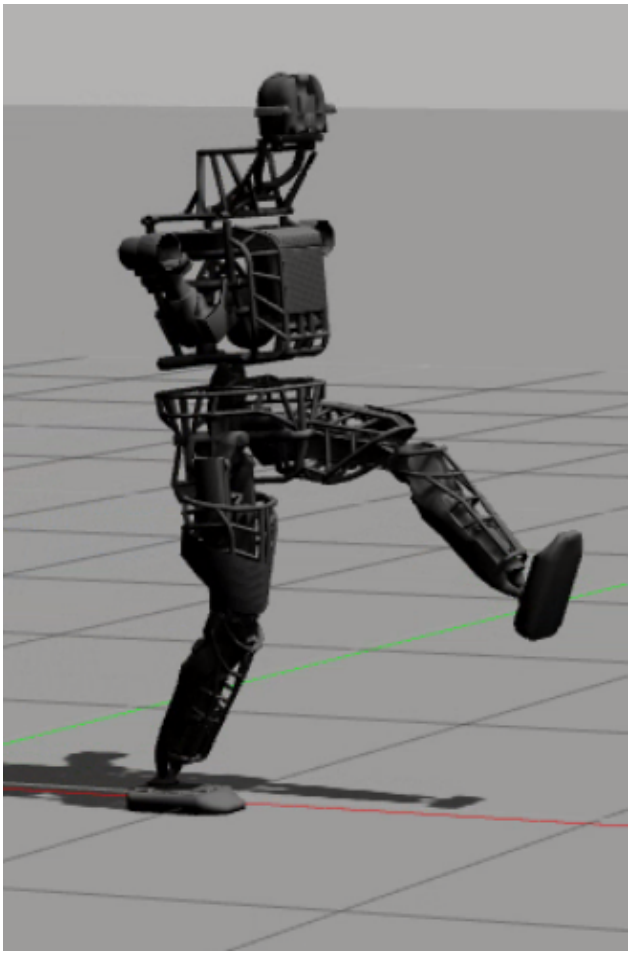


Figure 3.22.: Atlas model in Gazebo simulation environment.

With suitable α and β in (3.19), 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 SRL framework with the purpose that it can effectively learn from failures, so that in a later learning process similar dangerous maneuvers are avoided.

3.6.2. Safety and Learning Performance

In general, the exploration process of the reinforcement 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 SRL 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 p_t, which determines the aggressiveness of actions. An appropriate probability threshold p_t 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 behaviours.

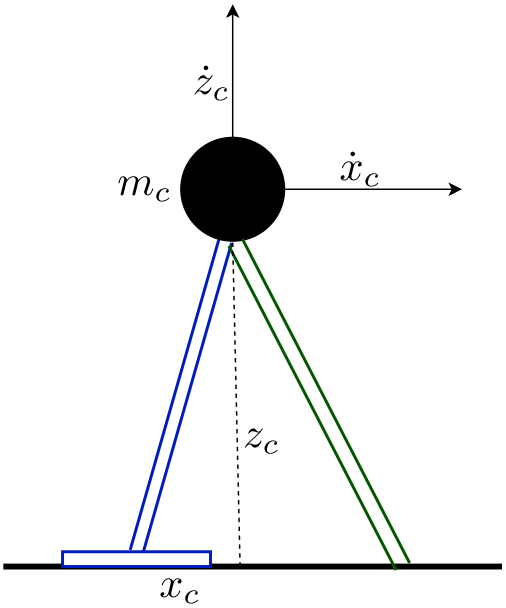


Figure 3.23.: A humanoid robot is simplified to an inverted pendulum model with respect to the CoM.

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 reinforcement learning algorithm [116]. By encouraging safe behaviours, 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.

3.6.3. Applications

The supervisory control strategy is compatible with arbitrary reinforcement learning algorithms, thus the proposed SRL 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 [117]. Note that, in this dissertation 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 [118] can be used in such cases to incorporate state constraints along with stability. The applicability of the proposed SRL 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 reinforcement 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 SRL framework aims to increase the safety when the reinforcement learning algorithm is improving the initial policy according to the real system behavior.

3.6.4. Limitations

One major limitation of our SRL 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 SRL 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 MOR, there is a trade-off 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.

3.7. Summary

In this chapter, we propose a general SRL framework for complex dynamical systems. A reinforcement learning-based controller is combined with a predefined corrective controller to ensure that during the exploration process of the learning algorithm, the safety of the dynamical system is maintained. This is achieved by using a supervisory control strategy that switches the actual applied actions from the learning-based controller to the corrective controller when the current system state is on the boundary of the estimated safe region. We utilize the concept of ROA to define the safe region in this chapter. Note that, other concepts, e.g., control invariant set or barrier function, can also be used, as long as a closed and control invariant safe region can be identified. To enable the practical implementation on complex dynamical systems, a simplified system that is found by physically inspired MOR is introduced to give estimates about the safe region. Considering the potential inaccuracy in the simplified system model, these safety estimates are presented in probabilistic form. The supervisor is then initialized by using the probabilistic estimates of safety obtained from the simplified system. For having a more reliable supervisor for the SRL framework, we also propose an online adaptation method to modify our beliefs about the safe region. Based on the belief function theory, two belief maps, the prior and the feedback belief maps, are

constructed accordingly. While the prior belief map uses the simplified system as the belief source, the feedback belief map is generated from the feedback data, i.e., the actual execution results of the corrective controller. These two belief maps are then combined via the weighted belief fusion to have more accurate probabilistic estimates of safety, which lead to a better performed supervisor for the SRL framework. Three examples are given to demonstrate the performance of the proposed SRL framework. First, a simple two-link inverted pendulum example is presented to explain in detail how the online adaptation method works. Second, a quadcopter flight control example shows the performance of the SRL framework on complex dynamical systems, both in simulations and in a real-world experiment. Third, a humanoid control task is employed to further illustrate the applicability of the SRL framework to different learning tasks. We believe that the proposed SRL framework is implementable on a wide range of dynamical systems, and it gives an insight about how to safely extend modern reinforcement learning methods to real-world control tasks.

One major limitation of the proposed SRL framework is that, the physically inspired MOR usually requires a thorough understanding about the system dynamics. Unfortunately, this is often not available for many practical control tasks. Hence in next chapter, we extend the SRL framework with a data-driven MOR technique that is used to find the simplified system model, such that the drawbacks of physically inspired MOR are addressed. Another possible direction for the future work is that, an effective way of tuning the parameters of the SRL framework is desired. Currently, the parameters are manually selected based on experience and a considerable amount of effort is required until a satisfying performance can be achieved. An automatic parameter tuning method could highly enhance the efficiency of the proposed SRL framework and therefore increase its applicability.

Safe Reinforcement Learning Based on Data-driven Model Order Reduction

4.

The SRL framework based on physically inspired MOR has one major limitation that, it requires a thorough understanding about the system dynamics such that representative physical features can be identified. Besides, multiple performance tests are often required before a satisfying simplified system can be found. As a result, the application of the SRL framework proposed in Chapter 3 is limited. To overcome these problems, we propose in this chapter a SRL framework that uses a novel data-driven approach to identify the simplified system model.

Inspired by transfer learning [90], we assume that an approximated system model of the complex dynamical system is available. Even though, inevitably, the approximated model displays discrepancies compared with the real system behavior, an initial estimate of safety can usually be obtained by simulating the approximated model. For example, while the dynamics of a real-world humanoid cannot be known perfectly, an approximated humanoid model can be constructed in simulation for making predictions. Hence, by simulating the system, we obtain training data that represents the safety of various original system states. However, as the state space is high-dimensional, it is infeasible to acquire a sufficient amount of training data to directly learn the safe region of the original system. To solve this problem, a data-driven approach that computes probabilistic similarities between each training data is proposed to first learn a low-dimensional representative safety feature of the complex dynamical system. Such a safety feature constitute the simplified system model and leads to a low-dimensional representation of the safe region, which is used as the starting point to SRL in the real system.

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

The remainder of this chapter is organized as follows: to better clarify our approach, an overview of the proposed approach is first given in Section 4.1. Then in Section 4.2, we explain in detail a data-driven method to derive a low-dimensional representation of the safe region. This is followed by the online adaptation method presented in Section 4.3, which is

used to update the identified low-dimensional representation. Two examples are provided in Section 4.4 to demonstrate the performance of the proposed approach. In Section 4.5, we discuss several properties of the approach, and Section 4.6 concludes this chapter.

4.1. Overview of the Approach

In this chapter, we consider the same supervisory control strategy as explained in Section 2.3 for constructing a SRL framework. Guided by the supervisor, the actual applied action during the learning process is switched between the learning-based controller $\pi(x)$ and the corrective controller $K(x)$. The safe region $\mathcal{S}$ is defined by using the ROA of the corrective controller $K(x)$ such as given in Definition 1. Note that, it is infeasible to employ physically inspired MOR when a thorough understanding about the system dynamics is not available, or there exists unknown part of the system dynamics or disturbance. We address this problem by adapting a data-driven MOR technique to find a simplified system model. To better explain the proposed approach, we present an overview of the approach in this section.

4.1.1. SRL for Complex Dynamical Systems

For realizing a feasible implementation of the SRL framework on complex dynamical system, a simplified system model is introduced to provide an approximation of the high-dimensional safe region of the original system. This is achieved by first mapping each original system state x to a simplified system state x_s through a state mapping $x_s = \Psi(x)$. The state mapping $x_s = \Psi(x)$ is chosen such that safe and unsafe states are separated in the simplified state space $\mathcal{X}_s$. Nevertheless, due to the order reduction, multiple original system states that have different safety properties can map to the same simplified state. Then, the safety of the original system state x is estimated by the safety of its corresponding simplified state x_s in a probabilistic form as

$$\mathbb{P}(x \in \mathcal{S}) = \Gamma(x_s)|_{x_s=\Psi(x)} \sim [0, 1] \tag{4.1}$$

where $\Gamma(x_s)$ is a function defined over the simplified state space $\mathcal{X}_s$ and is referred to as the *safety assessment function (SAF)* in this chapter. Not only does the SAF $\Gamma(x_s)$ encode information relating to the safety of the simplified state x_s, it also includes the uncertainty involved in making predictions for a high-dimensional state by using a low-dimensional reduction.

For a given SAF $\Gamma(x_s)$, the probability $\mathbb{P}(x \in \mathcal{S})$ depends only on the simplified state x_s. By using a probability threshold p_t, we thus obtain a low-dimensional representation of the safe region, denoted as $\mathcal{S}_l$, in the simplified state space $\mathcal{X}_s$

$$\mathcal{S}_l = \{x_s \in \mathcal{X}_s \mid \Gamma(x_s) > p_t\} \tag{4.2}$$

which works as an approximation of the high-dimensional safe region $\mathcal{S}$. Accordingly, the supervisor of the SRL framework that switches between the learning-based controller $\pi(x)$ and the corrective controller $K(x)$ is represented as

$$u = \begin{cases} \pi(x), & \text{if } t < t' \\ K(x), & \text{else} \end{cases} \tag{4.3}$$

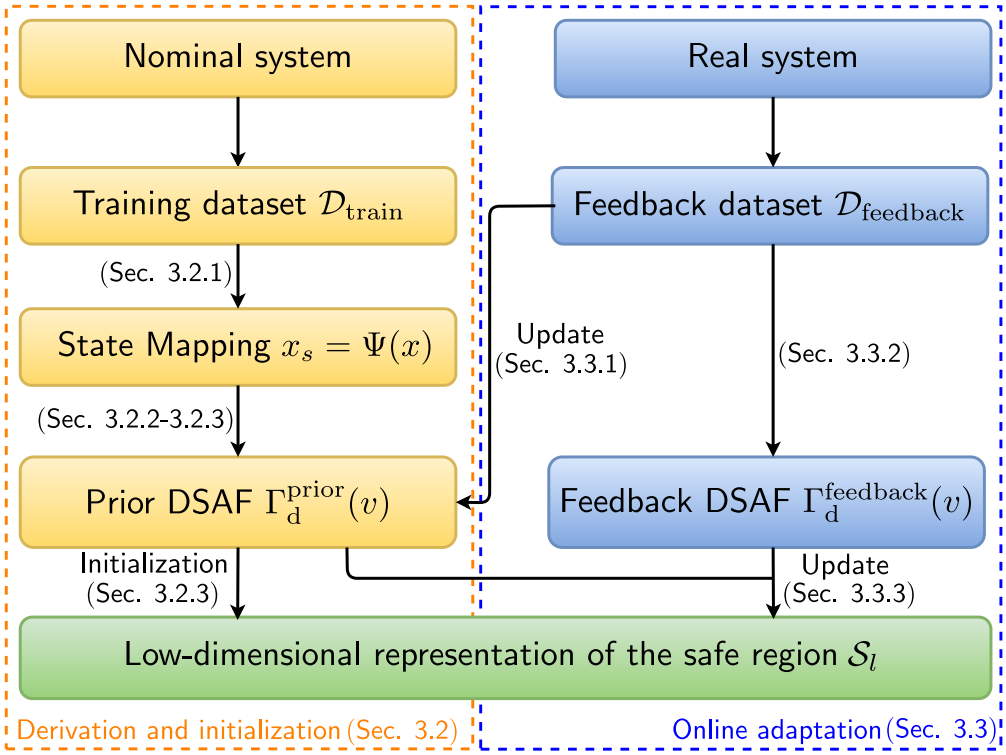


Figure 4.1.: Overview of the proposed approach. The low-dimensional representation $\mathcal{S}_l$ is initialized using the training dataset $\mathcal{D}_{\text{train}}$ obtained from the nominal system. Once we collect the feedback dataset $\mathcal{D}_{\text{feedback}}$ on the real system, the low-dimensional representation $\mathcal{S}_l$ is updated using the proposed online adaptation method.

where t' denotes the first time point at which the system state x has a corresponding simplified state x_s on the boundary of the low-dimensional representation of the safe region $\mathcal{S}_l$, i.e., $\mathbb{P}(x \in \mathcal{S}) = \Gamma(x_s) \leq p_t$. Apparently, the essential part of the SRL framework for complex dynamical systems is finding a SAF $\Gamma(x_s)$ that is as accurate as possible. To solve this problem, we demonstrate in this chapter how to efficiently identify the SAF $\Gamma(x_s)$ as well as the state mapping $x_s = \Psi(x)$ using a data-driven MOR technique, i.e., by learning a simplified system from the observed training data about safety.

4.1.2. SRL with Data-driven MOR

We propose in this chapter a novel data-driven approach to identity the SAF $\Gamma(x_s)$, together with a new online adaptation method to efficiently update the learned low-dimensional representation of the safe region $\mathcal{S}_l$.

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

$$\dot{x} = f(x) + g(x)u \tag{4.4}$$

which is the same as given in (2.19). The real system model is then given as

$$\dot{x} = f(x) + g(x)u + d(x) \tag{4.5}$$

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

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

Based on these facts, to construct a reliable low-dimensional representation of the safe region $\mathcal{S}_l$ for the real system, we propose the approach outlined in Fig. 4.1 (a complete work-flow is given in Fig. 4.2). It consists of two parts that solve the following two problems, respectively:

(i) How to derive and initialize the low-dimensional representation of the safe region $\mathcal{S}_l$ by using the nominal system model.

(ii) How to update the low-dimensional representation of the safe region $\mathcal{S}_l$ online with the observed real system behavior.

Part 1) Derivation and Initialization

Since no information about uncertainty $d(x)$ is available prior to the learning process, the corrective controller $K(x)$ is designed for the nominal system model (2.19). Although the safe region of the nominal system is unknown, its simulation is possible and delivers a dataset as follows.

Definition 5. *The training dataset of k_t training data is given as*

$$\mathcal{D}_{\text{train}} = \{D^1_{\text{train}}, D^2_{\text{train}}, \dots, D^{k_t}_{\text{train}}\}. \tag{4.6}$$

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

$$D^i_{\text{train}} = \{x^i_{\text{sim}}, s_{\text{sim}}(x^i_{\text{sim}}), \Phi_{\text{sim}}(t; x^i_{\text{sim}})\}. \tag{4.7}$$

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

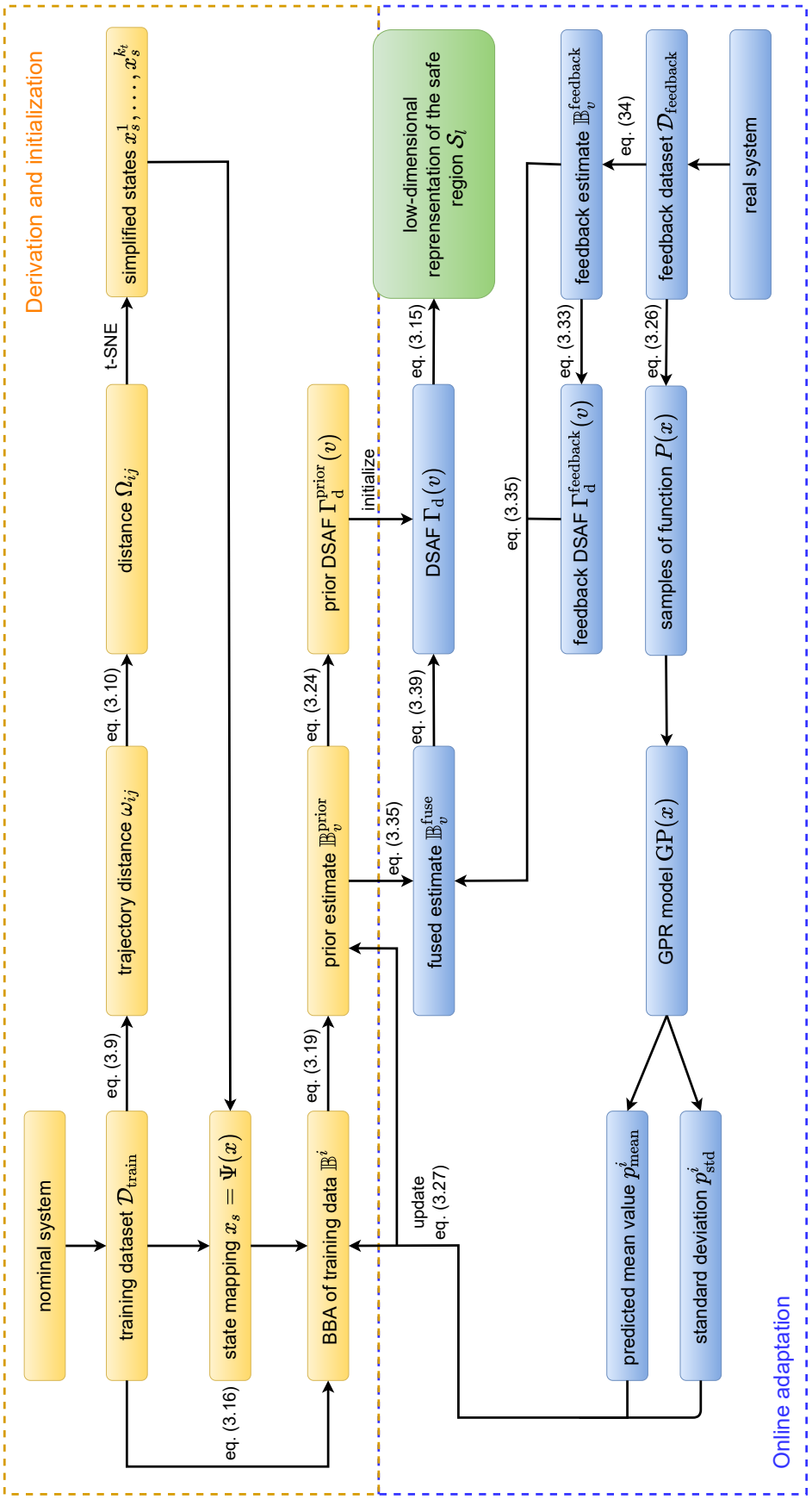


Figure 4.2.: Detailed work-flow of the proposed approach. The low-dimensional representation of the safe region $\mathcal{S}_l$ is initialized using the training dataset $\mathcal{D}_{\text{train}}$ and is updated using the feedback dataset $\mathcal{D}_{\text{feedback}}$.

A first low-dimensional representation of the safe region $\mathcal{S}_l$ is thus derived and initialized by using the training dataset $\mathcal{D}_{\text{train}}$. To do this, we first identify the state mapping $x_s = \Psi(x)$ using a data-driven method that computes the probabilistic similarity between each training data (see Section 4.2.1). Then to facilitate an efficient computation, we discretize the simplified state space $\mathcal{X}_s$ into grid cells and assign an index vector $v \in \mathbb{Z}_+^{n_s}$ to each grid cell. By assuming that the SAF $\Gamma(x_s)$ is constant in each grid cell, we thus obtain a discretized safety assessment function (DSAF) $\Gamma_{\text{d}}(v)$. A discretized low-dimensional representation of the safe region $\mathcal{S}_l$ is then given by introducing the probability threshold p_t to the DSAF $\Gamma_{\text{d}}(v)$ (see Section 4.2.2). To enable the implementation of the SRL framework on the real system, we also calculate an initial estimate of the DSAF $\Gamma_{\text{d}}(v)$, denoted as the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$, from the training dataset $\mathcal{D}_{\text{train}}$. It is then used as the initial low-dimensional representation of the safe region $\mathcal{S}_l$ (see Section 4.2.3). Further details of Part 1) are given in Section 4.2.

Part 2) Online Adaptation

After the initialization given in Part 1), we obtain an initial estimate of the low-dimensional representation of the safe region $\mathcal{S}_l$ that is identified by using the training dataset $\mathcal{D}_{\text{train}}$. However, due to the unknown part of the system dynamics $d(x)$, there is inevitably a mismatch between simulation and reality. In order to compensate for this mismatch, we update the obtained initial estimate of the low-dimensional representation of the safe region $\mathcal{S}_l$ by accounting for the real system behavior.

Each time the corrective controller $K(x)$, which is designed according to the nominal system, is activated during the learning process, we observe feedback data that reflects the real safe region, i.e., whether the corrective controller $K(x)$ is still able to keep the real system safe or not. The set of feedback data is defined as follows.

Definition 6. *The feedback dataset of k_f feedback data is given as*

$$\mathcal{D}_{\text{feedback}} = \{D_{\text{feedback}}^1, D_{\text{feedback}}^2, \ldots, D_{\text{feedback}}^{k_f}\}. \tag{4.8}$$

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

$$D_{\text{feedback}}^i = \{x_{\text{real}}^i, s_{\text{real}}(x_{\text{real}}^i), \Phi_{\text{real}}(t; x_{\text{real}}^i)\}. \tag{4.9}$$

While x_{real}^i, $s_{\text{real}}(x_{\text{real}}^i)$ and $\Phi_{\text{real}}(t; x_{\text{real}}^i)$ have the same meaning as in Definition 5, the subscript real *indicates here that the data is collected on the real system.*

Since collecting data on the real system, e.g., real-world robots, is usually expensive and time-consuming, in most cases the feedback dataset $\mathcal{D}_{\text{feedback}}$ has a limited size. Therefore, the estimate of the low-dimensional representation of the safe region $\mathcal{S}_l$ needs to be updated in a data-efficient manner. To achieve this, we propose an online adaptation method, as given in Section 4.3. It comprises three steps: First, we modify the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ by changing our confidence in its reliability using the feedback dataset $\mathcal{D}_{\text{feedback}}$ (see Section 4.3.1). Second, to fully utilize the valuable information contained in the feedback dataset $\mathcal{D}_{\text{feedback}}$, we generate another feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$ (see Section 4.3.2). Third, the two DSAFs are fused to obtain a more accurate DSAF $\Gamma_{\text{d}}(v)$, which is then used to update the low-dimensional representation $\mathcal{S}_l$ (see Section 4.3.3).

4.2. Learning a Low-dimensional Representation of the Safe Region

To derive the low-dimensional representation of the safe region $\mathcal{S}_l$, two components have to be determined: the state mapping $x_s = \Psi(x)$, which gives the low-dimensional safety feature, and the SAF $\Gamma(x_s)$, which predicts the safety of original system states. In this section, we present a data-driven method for identifying the low-dimensional representation of the safe region $\mathcal{S}_l$. It utilizes a technique called t-distributed stochastic neighbor embedding (t-SNE) [119], which is originally proposed for visualizing high-dimensional data. Through measuring the similarity between each high-dimensional data point, t-SNE defines a two- or three-dimensional data point such that similar high-dimensional data points are represented by nearby low-dimensional data points with high probability. Details are presented in the following subsections.

4.2.1. Identifying the State Mapping with t-SNE

To identify the state mapping $x_s = \Psi(x)$, we first find the realization of the low-dimensional safety feature, i.e., the values of simplified states $x_s^1, \ldots, x_s^{k_t}$, that best corresponds with the training dataset $\mathcal{D}_{\text{train}}$ by revising t-SNE. Unlike the original t-SNE, which uses Euclidean distance between each pair of high-dimensional data points as the metric for measuring similarity, we propose a new similarity metric that considers similarity and safety at the same time. The reason for this is that, since our purpose is to construct the low-dimensional representation of the safe region $\mathcal{S}_l$, we are more interested in safety rather than just distance.

The general motivation for determining the simplified state x_s is that the safe and unsafe original system states x should be separated in the simplified state space $\mathcal{X}_s$. Since, in this dissertation, the safe region is defined with respect to the ROA, the trajectories of safe initial states will converge to the origin, while unsafe initial states will have divergent trajectories. Hence, if two original system states x have similar trajectories under the corrective controller $K(x)$, then ideally they should also have nearby corresponding simplified states x_s (see Fig. 4.3). Based on this, we first calculate the pairwise trajectory distance ω_{ij} between the i-th and j-th training data, using dynamic time warping (DTW) as

$$\omega_{ij} = \text{dtw}(\Phi_{\text{sim}}(t; x_{\text{sim}}^i), \Phi_{\text{sim}}(t; x_{\text{sim}}^j)) \tag{4.10}$$

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

Remark 7. *Besides DTW, other trajectory distance measures, e.g., Fréchet distance [120], can be used in (4.10). Changing the distance metric does not affect the applicability of the proposed approach. However, DTW turns out to be a more suitable metric for trajectories of the dynamical systems we investigated.*

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

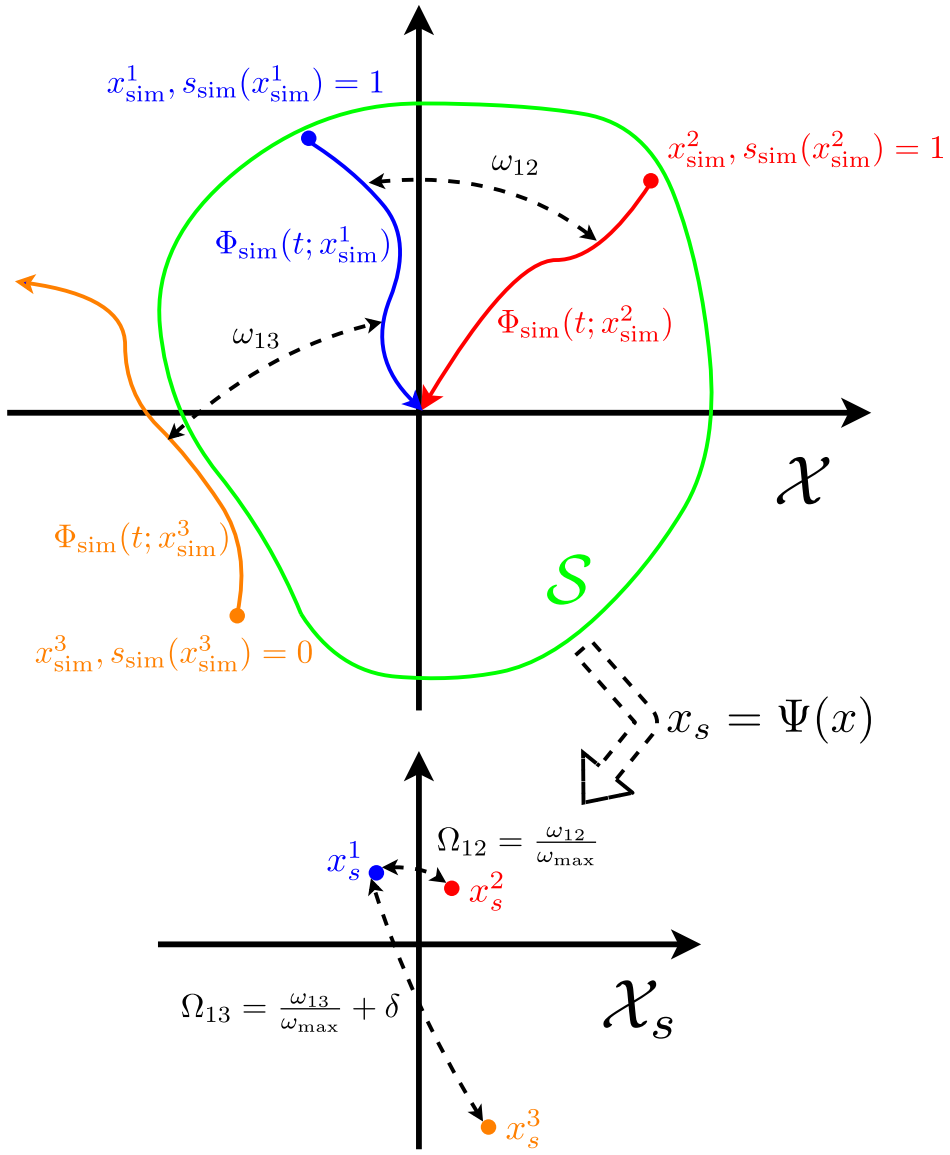


Figure 4.3.: The distances Ω_{12} and Ω_{13} are computed for three training data D^1_{train}, D^2_{train}, D^3_{train} using the trajectory distances ω_{12}, ω_{13} and the safety labels $s_{\text{sim}}(x^1_{\text{sim}})$, $s_{\text{sim}}(x^2_{\text{sim}})$, $s_{\text{sim}}(x^3_{\text{sim}})$. Based on these distances, t-SNE calculates the values of corresponding simplified states x_s, where similar and dissimilar training data are modeled by nearby and distant simplified states, respectively.

distance Ω_{ij} between the i-th and j-th training data as

$$\Omega_{ij} = \begin{cases} \dfrac{\omega_{ij}}{\omega_{\max}} + \delta, & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) \neq s_{\text{sim}}(x^j_{\text{sim}}) \\ \dfrac{\omega_{ij}}{\omega_{\max}}, & \text{if } s_{\text{sim}}(x^i_{\text{sim}}) = s_{\text{sim}}(x^j_{\text{sim}}) \end{cases} \tag{4.11}$$

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

In our experiments, we find that a small value of δ is sufficient for providing a satisfying

result of t-SNE (in this chapter, for example, we use $\delta = 0.01$). A large value of δ, in contrast, may lead to information contained in trajectories being ignored, which can reduce the representation power of the learned simplified states x_s. A sensitivity analysis of the parameter δ is provided in Appendix B.

After computing the distance Ω_{ij} between each pair of training data, we apply t-SNE on the training dataset $\mathcal{D}_{\text{train}}$ to derive a realization of the low-dimensional safety feature. To do this, we modify the conditional probability $p_{j|i}$ of t-SNE [119] using the distance Ω_{ij} as

$$p_{j|i} = \frac{\exp(-\Omega_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\Omega_{ik}^2/2\sigma_i^2)} \tag{4.12}$$

where σ_i is the variance of the Gaussian distribution that is centered on the state x_{sim}^i. The remaining computations are the same as in t-SNE. The main steps involved in performing t-SNE are given in Appendix C.

Using t-SNE, we obtain the values of simplified states $x_s^1, \ldots, x_s^{k_t}$ that correspond to the training dataset $\mathcal{D}_{\text{train}}$ as an initial realization of the low-dimensional safety feature. Such a realization models similar training data with nearby simplified states, e.g., x_s^1 and x_s^2 in Fig. 4.3, and dissimilar training data with distant simplified states, e.g., x_s^1 and x_s^3 in Fig. 4.3. In general, the simplified state x_s is chosen to be two- or three-dimensional, i.e., $x_s \in \mathbb{R}^{n_s}$ with $n_s = 2$ or $n_s = 3$. In this chapter, we set $n_s = 2$.

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

Remark 8. *Different forms of function approximator, for instance, a Gaussian process, can be used to describe the state mapping $x_s = \Psi(x)$. The selection of function approximator depends mainly on the available number of training data.*

Due to the approximation error in the function approximator, some original system states x may have slightly different values in their simplified states x_s when comparing the initial realization obtained from t-SNE with the one computed from the learned state mapping $x_s = \Psi(x)$ (for an example, see the simulations in Section 4.4.1 and in particular Fig. 4.8). Hence, to reduce the influence of this issue on deriving the low-dimensional representation of the safe region $\mathcal{S}_l$, we compute the values of simplified states $x_s^1, \ldots, x_s^{k_t}$ once again with the learned state mapping. This final realization of the low-dimensional safety feature is then used for formulating the SAF $\Gamma(x_s)$, which is detailed in next subsection.

4.2.2. Defining the DSAF with Belief Function Theory

Once the state mapping $x_s = \Psi(x)$ is determined, we are able to calculate a SAF $\Gamma(x_s)$ that predicts whether a given original system state x is safe or not. As for now, all available information regarding safety is contained in the training data, we will use the training dataset $\mathcal{D}_{\text{train}}$ for computing the SAF $\Gamma(x_s)$. However, due to the limited size of the training data,

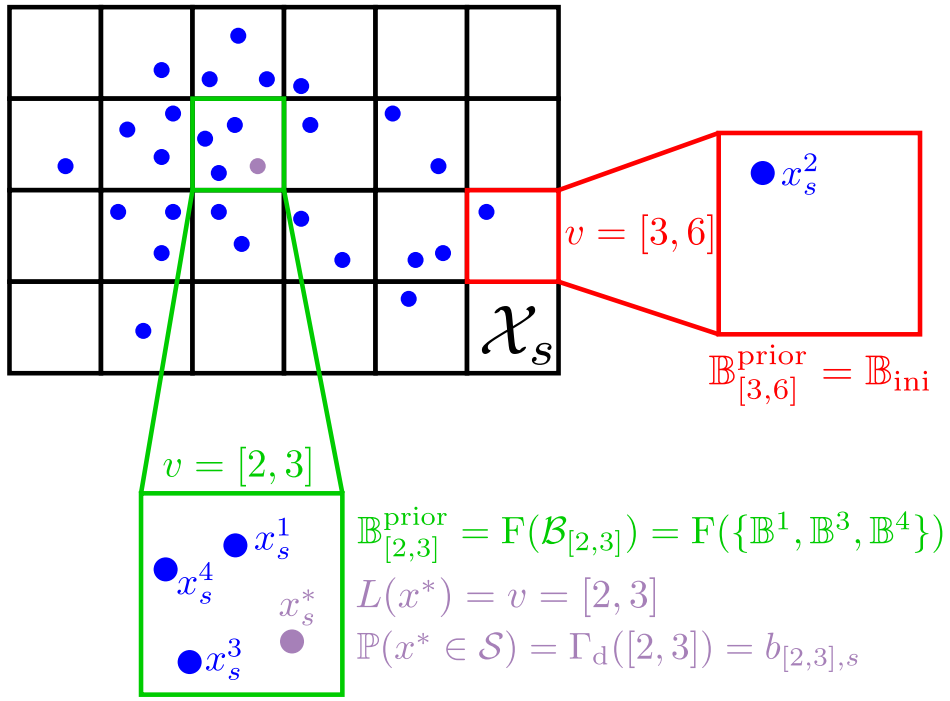


Figure 4.4.: The simplified state space $\mathcal{X}_s$ is discretized into grid cells. The location of each grid cell is indicated by the index vector v. The safety of a new original system state, e.g. x^*, is estimated by way of the corresponding belief mass as $\mathbb{P}(x^* \in \mathcal{S}) = \Gamma_{\text{d}}([2, 3]) = b_{[2,3],s}$, where $L(x^*) = v = [2, 3]$. The prior estimate $\mathbb{B}^{\text{prior}}_v$ of an index vector v is either obtained by fusing all BBAs within the set $\mathcal{B}_v$, e.g., $\mathbb{B}^{\text{prior}}_{[2,3]} = \text{F}(\mathcal{B}_{[2,3]})$, or set to an initial estimate, e.g., $\mathbb{B}^{\text{prior}}_{[3,6]} = \mathbb{B}_{\text{ini}}$.

it is computationally infeasible to construct the SAF $\Gamma(x_s)$ over the continuous simplified state space $\mathcal{X}_s$. Therefore, we discretize the simplified state space $\mathcal{X}_s$.

The range of the simplified state space $\mathcal{X}_s$ is determined by the maximum and minimum values of the simplified states $x_s^1, \ldots, x_s^{kt}$ in each dimension. We then discretize the simplified state space $\mathcal{X}_s$ into grid cells with a predefined step size. Each grid cell is assigned an index vector $v \in \mathbb{Z}_+^2$ to indicate its position in the simplified state space $\mathcal{X}_s$; for example, $v = [2, 3]$ refers to the grid cell that is located at the second row and third column (see Fig. 4.4). A same locating function $L(x)$ as in Definition 3 is also introduced here.

Definition 7. *By locating the simplified state $x_s = \Psi(x)$ for an original system state x in the simplified state space $\mathcal{X}_s$, the locating function $L(x)$ returns the index vector v of the grid cell that it belongs to.*

By assuming that the SAF $\Gamma(x_s)$ is constant in each grid cell, we obtain a DSAF $\Gamma_{\text{d}}(v)$ that we will have to define. Then, instead of using the simplified state x_s, the safety of an original system state x is estimated by using the index vector v as

$$\mathbb{P}(x \in \mathcal{S}) = \Gamma_{\text{d}}(v)|_{v=L(x)} \sim [0, 1]. \tag{4.13}$$

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

cases, infeasible to acquire a sufficient amount of data to derive an accurate estimate. To solve this problem, we use the belief function theory [108] (see Section 3.3.1) to describe the DSAF $\Gamma_{\mathrm{d}}(v)$, where the uncertainty caused by insufficiency in the data amount is considered by a subjective probability [109]. We therefore define a BBA $\mathbb{B}_v$ separately for each index vector v as follows.

Definition 8. *The BBA $\mathbb{B}_v$ for an index vector v is given as*

$$\mathbb{B}_v = (b_{v,s}, b_{v,u}, \sigma_v) \tag{4.14}$$

which represents the belief about the value of the DSAF $\Gamma_{\mathrm{d}}(v)$ for the index vector v. The belief masses $b_{v,s}$ and $b_{v,u}$ and the subjective uncertainty σ_v have the same meaning as in Definition 4.

Hence the DSAF $\Gamma_{\mathrm{d}}(v)$ is given by the belief masses $b_{v,s}$ of the corresponding BBAs $\mathbb{B}_v$ as

$$\Gamma_{\mathrm{d}}(v) = b_{v,s}. \tag{4.15}$$

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

$$\mathcal{S}_l = \{v \mid \Gamma_{\mathrm{d}}(v) = b_{v,s} > p_t\} \tag{4.16}$$

where p_t is the predefined probability threshold. In the next subsection, we explain how to initialize the DSAF $\Gamma_{\mathrm{d}}(v)$ from the training dataset $\mathcal{D}_{\mathrm{train}}$ so as to enable the application of the SRL framework on the real system.

4.2.3. Initializing the DSAF from Training Data

Since each training data provides information on the value of the DSAF $\Gamma_{\mathrm{d}}(v)$, the low-dimensional representation of the safe region $\mathcal{S}_l$ is initialized using the training dataset $\mathcal{D}_{\mathrm{train}}$. This is done by considering each training data as a belief source and then fuse all beliefs together. The result is a prior estimate of the DSAF $\Gamma_{\mathrm{d}}(v)$, which is used to initialize the SRL framework.

We first formulate a BBA for each training data as follows.

Definition 9. *The BBA $\mathbb{B}^i$ obtained from the i-th training data D^i_{train} is defined as*

$$\mathbb{B}^i = (b^i_s, b^i_u, \sigma^i). \tag{4.17}$$

It represents the belief about the value of the DSAF $\Gamma_{\mathrm{d}}(v)$ for the index vector $v = L(x^i_{\mathrm{sim}})$, where the belief source is the i-th training data. b^i_s, b^i_u and σ^i are equivalent to $b_{v,s}$, $b_{v,u}$ and σ_v, with the superscript i indicating the index of the training data.

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

$$\mathbb{B}^i = \begin{cases} (1 - \sigma_{\mathrm{ini}}, 0, \sigma_{\mathrm{ini}}), & \text{if } s_{\mathrm{sim}}(x^i_{\mathrm{sim}}) = 1 \\ (0, 1 - \sigma_{\mathrm{ini}}, \sigma_{\mathrm{ini}}), & \text{if } s_{\mathrm{sim}}(x^i_{\mathrm{sim}}) = 0 \end{cases} \tag{4.18}$$

where $i = 1, \ldots, k_t$. Since no information about the unknown term $d(x)$ is available prior to the learning process on the real system, the initial subjective uncertainties are chosen to be

the same for all BBAs. Later in the online adaptation method, the subjective uncertainties are updated by using the feedback data to realize more accurate safety estimates.

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

$$\mathcal{B}_v = \{\mathbb{B}^i \mid L(x^i_{\text{sim}}) = v\}. \tag{4.19}$$

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

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

$$\mathbb{B}_v^{\text{prior}} = (b_{v,s}^{\text{prior}}, b_{v,u}^{\text{prior}}, \sigma_v^{\text{prior}}) = \begin{cases} \text{F}(\mathcal{B}_v), & \text{if } k_v \geq k_{\min} \\ \mathbb{B}_{\text{ini}}, & \text{else} \end{cases} \tag{4.20}$$

where $\mathbb{B}_{\text{ini}}$ is an initial estimate that represents our guess about the BBA $\mathbb{B}_v$ when no training data is available (see Fig. 4.4). $\text{F}(\cdot)$ is a fusion operation among the set $\mathcal{B}_v$, which is the weighted belief fusion for multiple belief sources and is defined according to [110] as

$$b_{v,s}^{\text{prior}} = \frac{\sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} b_s^i (1-\sigma^i) \prod\limits_{\substack{\mathbb{B}^j \in \mathcal{B}_v \\ i \neq j}} \sigma^j}{\left(\sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} \prod\limits_{\substack{\mathbb{B}^j \in \mathcal{B}_v \\ i \neq j}} \sigma^j \right) - k_v \prod\limits_{\mathbb{B}^i \in \mathcal{B}_v} \sigma^i} \tag{4.21}$$

$$b_{v,u}^{\text{prior}} = \frac{\sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} b_u^i (1-\sigma^i) \prod\limits_{\substack{\mathbb{B}^j \in \mathcal{B}_v \\ i \neq j}} \sigma^j}{\left(\sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} \prod\limits_{\substack{\mathbb{B}^j \in \mathcal{B}_v \\ i \neq j}} \sigma^j \right) - k_v \prod\limits_{\mathbb{B}^i \in \mathcal{B}_v} \sigma^i} \tag{4.22}$$

$$\sigma_v^{\text{prior}} = \frac{\left(k_v - \sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} \sigma^i \right) \prod\limits_{\mathbb{B}^i \in \mathcal{B}_v} \sigma^i}{\left(\sum\limits_{\mathbb{B}^i \in \mathcal{B}_v} \prod\limits_{\substack{\mathbb{B}^j \in \mathcal{B}_v \\ i \neq j}} \sigma^j \right) - k_v \prod\limits_{\mathbb{B}^i \in \mathcal{B}_v} \sigma^i}. \tag{4.23}$$

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

$$b_{v,s}^{\text{prior}} + b_{v,u}^{\text{prior}} + \sigma_v^{\text{prior}} = 1 \tag{4.24}$$

and $b_{v,s}^{\text{prior}}$, $b_{v,u}^{\text{prior}}$, σ_v^{prior} all lie within the interval $[0, 1]$.

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

$$\Gamma_{\text{d}}^{\text{prior}}(v) = b_{v,s}^{\text{prior}} \tag{4.25}$$

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

4.3. Online Adaptation of the Safe Region

In the early learning phase on the real system, together with the identified state mapping $x_s = \Psi(x)$, the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ allows the supervisor to perform a rough estimate of the safety of an original system state. During the learning process, the feedback data is used to update the DSAF $\Gamma_{\text{d}}(v)$ to achieve more accurate safety estimates. Note that each time the corrective controller $K(x)$ is activated for the real system, we obtain new feedback data. Hence the size of the feedback dataset $\mathcal{D}_{\text{feedback}}$ increases incrementally during the learning process. For simplicity, we consider the feedback dataset $\mathcal{D}_{\text{feedback}}$ of size k_f in this section.

Each update iteration of the DSAF $\Gamma_{\text{d}}(v)$ consists of three steps. First, we modify the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ by revising our beliefs about the reliability of the training data (see Section 4.3.1). Second, we compute a feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$ using the feedback data (see Section 4.3.2). Third, the updated DSAF $\Gamma_{\text{d}}(v)$ is obtained by fusing the prior and feedback DSAFs (see Section 4.3.3). Details of the online adaptation method are given in the following.

4.3.1. Update of the Prior DSAF with Feedback Data

Due to the simulation-to-reality gap, the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ derived from the training data has discrepancy compared to the real system behavior. For compensating such a discrepancy, we use feedback data to update our beliefs about the reliability of each training data, which then results in an updated and more accurate prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$.

In the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$, the uncertainty caused by the unknown term $d(x)$ is represented by the subjective uncertainty σ^i of each BBA $\mathbb{B}^i$. Hence, the update of the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ is done by modifying the subjective uncertainties using new information given by feedback data. As a prerequisite for relating the training and the feedback data, we assume that the following assumption holds, which states that original system states that are in close proximity to each other most probably have similar safety properties.

Assumption 5. *The probability $\mathbb{P}(s_{\text{real}}(x^1) = s_{\text{real}}(x^2))$ that two original system states x^1 and x^2 have the same safety property on the real system is inversely proportional to their Euclidean distance in the original state space $||x^1 - x^2||$.*

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

$$P(x) = \mathbb{P}(s_{\text{sim}}(x) = s_{\text{real}}(x)) \sim [0, 1]. \tag{4.26}$$

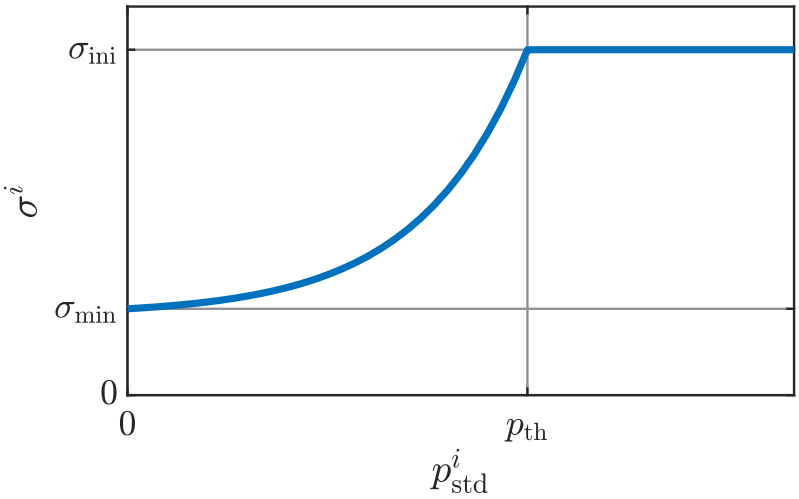


Figure 4.5.: As given in (4.29), the subjective uncertainty σ^i in the BBA B^i of the i-th training data is determined using the corresponding standard deviation p^i_{std} obtained from the GPR model $\text{GP}(x)$.

It represents the probability that for a given original system state x, its safety label $s_{\text{sim}}(x)$ obtained with the nominal system is the same as the safety label $s_{\text{real}}(x)$ obtained with the real system. Then, according to Assumption 5, if we observe an original system state x that has the same safety property both in simulation and in reality, it is likely that other original system states that are close to the observed state will also show the same safety property.

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

$$P(x^i_{\text{real}}) = \begin{cases} 1, & \text{if } s_{\text{sim}}(x^i_{\text{real}}) = s_{\text{real}}(x^i_{\text{real}}) \\ 0, & \text{if } s_{\text{sim}}(x^i_{\text{real}}) \neq s_{\text{real}}(x^i_{\text{real}}) \end{cases} \tag{4.27}$$

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

Remark 9. *If the real system is a real-world dynamical system, then it is usually difficult to test the corrective controller $K(x)$ with arbitrary initial original system states x in reality, since there is a high risk of encountering unsafe behaviors. However in contrast, the simulation can be initialized with any original system state x_{real} contained in the feedback data, which then makes it possible to approximate the function $P(x)$.*

The trained GPR model $\text{GP}(x)$ is then used to update the BBA $\mathbb{B}^i$. The general motivation is that, we decrease the subjective uncertainty σ^i if we are confident about the reliability of this training data. Hence for the i-th training data, we compute a predicted mean value of the function $P(x^i_{\text{sim}})$, denoted as p^i_{mean}, from the GPR model $\text{GP}(x)$, along with a corresponding standard deviation p^i_{std} of the predicted value. Since a low value of the standard deviation

p^i_{std} means we have observed enough feedback data to make a reliable prediction, we only update the BBA $\mathbb{B}^i$ if the standard deviation p^i_{std} is smaller than a predefined threshold p_{th}

$$\mathbb{B}^i = \begin{cases} (p^i_{\text{mean}}(1-\sigma^i), (1-p^i_{\text{mean}})(1-\sigma^i), \sigma^i), \\ \qquad \text{if } p^i_{\text{std}} \leq p_{\text{th}} \text{ and } s_{\text{sim}}(x^i_{\text{sim}}) = 1 \\ ((1-p^i_{\text{mean}})(1-\sigma^i), p^i_{\text{mean}}(1-\sigma^i), \sigma^i), \\ \qquad \text{if } p^i_{\text{std}} \leq p_{\text{th}} \text{ and } s_{\text{sim}}(x^i_{\text{sim}}) = 0 \end{cases} \tag{4.28}$$

with the new subjective uncertainty σ^i calculated as

$$\sigma^i = \frac{\sigma_{\text{ini}} - \sigma_{\text{min}}}{\gamma^{p_{\text{th}}} - 1}(\gamma^{p^i_{\text{std}}} - 1) + \sigma_{\text{min}} \tag{4.29}$$

where σ_{ini} is the same initial subjective uncertainty as that given in (4.18) (see Fig. 4.5 for a graphical representation of (4.29)). BBAs $\mathbb{B}^i$ with $p^i_{\text{std}} > p_{\text{th}}$ remain unchanged, as in (4.18). Such an update of the BBA $\mathbb{B}^i$ considers the predicted value of the function $P(x^i_{\text{sim}})$ and the reliability of this prediction at the same time.

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

Note that for the same training data, the relationship between the standard deviation p^i_{std} and the threshold p_{th} can change during the learning process. For example, we might obtain $p^i_{\text{std}} \leq p_{\text{th}}$ in the current update iteration, but in the next update iteration it changes to $p^i_{\text{std}} > p_{\text{th}}$. This happens primarily when we first observe a safe original system state but followed by a nearby unsafe state, such that the safety of the states in between these two observed states becomes uncertain. In such cases, we set the BBA $\mathbb{B}^i$ back to the initial BBA given in (4.18).

Once the BBAs $\mathbb{B}^i$ of all training data have been updated with the up-to-date feedback dataset $\mathcal{D}_{\text{feedback}}$, the prior estimate $\mathbb{B}^{\text{prior}}_v$ for each index vector v is recomputed using (4.20). This results in an updated prior DSAF $\Gamma^{\text{prior}}_{\text{d}}(v)$, which is used later for revising the DSAF $\Gamma_{\text{d}}(v)$.

4.3.2. Feedback DSAF

The feedback data contain the information about the real safety properties of different original system states x. On the one hand, this information is used to estimate the reliability of training data such as given in the update process of the prior DSAF $\Gamma^{\text{prior}}_{\text{d}}(v)$. On the other hand, the feedback data also provides new information about the actual safe region of the real system. To fully utilize this valuable information, we construct an additional DSAF, denoted as the feedback DSAF $\Gamma^{\text{feedback}}_{\text{d}}(v)$, using the feedback dataset $\mathcal{D}_{\text{feedback}}$.

As the amount of data is insufficient, we also consider the estimate obtained from the feedback data as a subjective probability [89]. Then, as with the prior estimate $\mathbb{B}^{\text{prior}}_v$, we

formulate another estimate of the BBA $\mathbb{B}_v$ for each index vector v as

$$\mathbb{B}_v^{\text{feedback}} = (b_{v,s}^{\text{feedback}}, b_{v,u}^{\text{feedback}}, \sigma_v^{\text{feedback}}) \tag{4.30}$$

which is referred to as the feedback estimate $\mathbb{B}_v^{\text{feedback}}$.

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

$$b_{v,s}^{\text{feedback}} = \frac{k_{v,s}}{k_{v,s} + k_{v,u}}(1 - \sigma_v^{\text{feedback}}) \tag{4.31}$$

$$b_{v,u}^{\text{feedback}} = \frac{k_{v,u}}{k_{v,s} + k_{v,u}}(1 - \sigma_v^{\text{feedback}}) \tag{4.32}$$

$$\sigma_v^{\text{feedback}} = \alpha \cdot e^{-\beta(k_{v,s}+k_{v,u}-1)}. \tag{4.33}$$

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

Using the feedback estimate $\mathbb{B}_v^{\text{feedback}}$, we obtain the following feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$

$$\Gamma_{\text{d}}^{\text{feedback}}(v) = b_{v,s}^{\text{feedback}} \tag{4.34}$$

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

4.3.3. Fusion of Prior and Feedback DSAFs

The prior and feedback DSAFs both provide beliefs about safety by using different datasets as their belief source. To obtain a more accurate estimate of the DSAF $\Gamma_{\text{d}}(v)$, we fuse these two functions using weighted belief fusion as given in (4.21-4.23). This leads to a fused estimate $\mathbb{B}_v^{\text{fuse}}$ for each index vector v

$$\mathbb{B}_v^{\text{fuse}} = (b_{v,s}^{\text{fuse}}, b_{v,u}^{\text{fuse}}, \sigma_v^{\text{fuse}}) \tag{4.35}$$

which is computed as

$$\mathbb{B}_v^{\text{fuse}} = \begin{cases} \text{F}(\{\mathbb{B}_v^{\text{prior}}, \mathbb{B}_v^{\text{feedback}}\}), & \text{if } \mathbb{B}_v^{\text{feedback}} \neq \mathbb{B}_\varnothing \\ \mathbb{B}_v^{\text{prior}}, & \text{if } \mathbb{B}_v^{\text{feedback}} = \mathbb{B}_\varnothing. \end{cases} \tag{4.36}$$

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

The fused estimate $\mathbb{B}_v^{\text{fuse}}$ fulfills the following property, which is the same as Proposition 1

Proposition 2. *If the number of feedback data approaches infinity, the fused estimate $\mathbb{B}_v^{\text{fuse}}$ becomes the actual probabilities, and the prior estimate $\mathbb{B}_v^{\text{prior}}$ has no effect in making safety estimates.*

Proof. Proposition 2 is justified by the following equations

$$\lim_{k_{v,s}+k_{v,u}\to\infty} b_{v,s}^{\text{fuse}} = b_{v,s}^{\text{feedback}} \tag{4.37}$$

$$\lim_{k_{v,s}+k_{v,u}\to\infty} b_{v,u}^{\text{fuse}} = b_{v,u}^{\text{feedback}} \tag{4.38}$$

$$\lim_{k_{v,s}+k_{v,u}\to\infty} \sigma_v^{\text{fuse}} = \sigma_v^{\text{feedback}} = 0 \tag{4.39}$$

which are obtained by simplifying (4.21-4.23) with the set $\{\mathbb{B}_v^{\text{prior}}, \mathbb{B}_v^{\text{feedback}}\}$. □

Considering computational efficiency, the update of the DSAF $\Gamma_{\text{d}}(v)$ is generally performed once when every k_u feedback data is obtained, where the value of k_u is selected according to the actual learning task. In each update iteration (indexed by number N, see Section 4.4.1), we first use the up-to-date feedback dataset $\mathcal{D}_{\text{feedback}}$ to update the prior DSAF $\Gamma_{\text{d}}^{\text{prior}}(v)$ and to construct the feedback DSAF $\Gamma_{\text{d}}^{\text{feedback}}(v)$. Then, the fused estimate $\mathbb{B}_v^{\text{fuse}}$ is computed from these two functions for each index vector v. The updated DSAF $\Gamma_{\text{d}}(v)$ is thus obtained using the fused estimate $\mathbb{B}_v^{\text{fuse}}$ as

$$\Gamma_{\text{d}}(v) = b_{v,s}^{\text{fuse}} \tag{4.40}$$

which also gives the latest low-dimensional representation of the safe region $\mathcal{S}_l$ according to (4.16). With further feedback data, the DSAF $\Gamma_{\text{d}}(v)$ becomes more accurate and more reliable safety estimates are obtained.

4.4. Experimental Results

In this section, we present two examples, a quadcopter and a humanoid, to demonstrate the performance of using the proposed approach for identifying the low-dimensional representation of the safe region $\mathcal{S}_l$.

4.4.1. Quadcopter Example

Experimental setup

We simulate the quadcopter using the system dynamics given in [121] with MATLAB Simulink[1] (Version R2019b) running on a laptop powered by an Intel i7-7700HQ CPU. The 12-dimensional system state is defined as $x = [p_g, \theta_g, v_b, \omega_b]^T$, where $p_g = [p_x, p_y, p_z]^T$ and $\theta_g = [\theta_r, \theta_p, \theta_y]^T$ are the linear and angular positions defined in the ground frame, $v_b = [v_x, v_y, v_z]^T$ and $\omega_b = [\omega_r, \omega_p, \omega_y]^T$ are the linear and angular velocities defined in the

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

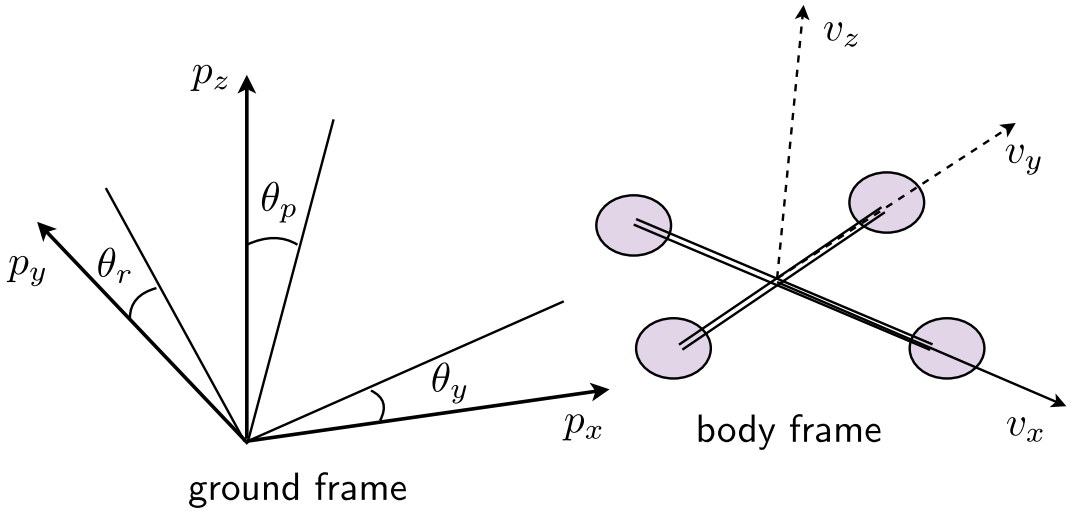


Figure 4.6.: The system state x of a quadcopter is defined using the ground frame and the body frame.

body frame (see Fig. 4.6). The control input u consists of the four motor speeds of the quadcopter. For the nominal system model, we set the mass of the quadcopter to $m = 1$ kg and the maximal lifting force to $f = 200$ N. The safety of a given state x is determined by simulating the controlled dynamics with the corrective control $K(x)$ that starts in initial state x, and checking if the controller is able to successfully drive the quadcopter back to a hovering state without crashing. In this example, we use the PID controller given in [121] as the corrective controller $K(x)$. It stabilizes the quadcopter's height as well as its roll, pitch and yaw rotations. The coefficients of the PID controller are: $K_{P,h} = 1.5$, $K_{I,h} = 0$, $K_{D,h} = 2.5$ for the height control, and $K_{P,r} = K_{P,p} = K_{P,y} = 6$, $K_{I,r} = K_{I,p} = K_{I,y} = 0$, $K_{D,r} = K_{D,p} = K_{D,y} = 1.75$ for the roll, pitch and yaw rotations control, respectively.

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

Identifying the low-dimensional representation of the safe region

The initial realization of the low-dimensional safety feature, i.e., the values of simplified states $x_s^1, \ldots, x_s^{k_t}$, obtained from t-SNE is given in Fig. 4.7. We use $\delta = 0.01$ in (4.11) and set the perplexity and tolerance of t-SNE (see Appendix C) to 40 and $1e^{-4}$, respectively. The result shows that the safe and unsafe original system states are clearly separated in the two-dimensional simplified state space $\mathcal{X}_s \subseteq \mathbb{R}^2$.

The state mapping $x_s = \Psi(x)$ is represented by a two-layer neural network with 128 neurons in each layer, which is trained using the initial realization of simplified states $x_s^1, \ldots, x_s^{k_t}$ and the set of original system states $\{x_{\text{sim}}^1, \ldots, x_{\text{sim}}^{k_t}\}$. By recomputing the outputs of the learned neural network, we obtain the final realization of the low-dimensional safety feature, i.e., the values of the simplified states $x_s^1, \ldots, x_s^{k_t}$, given in Fig. 4.8. Due to approximation

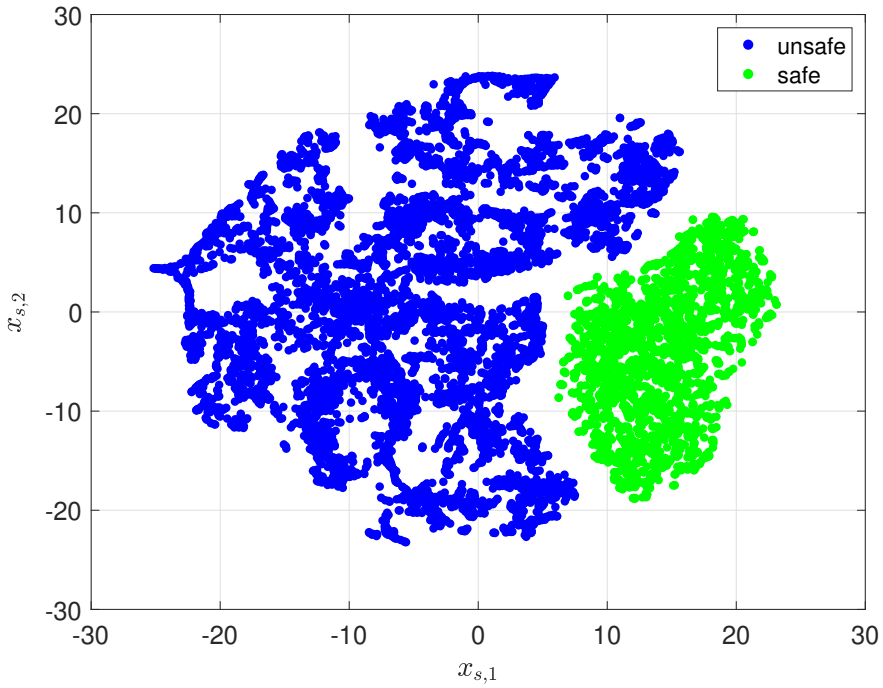


Figure 4.7.: The initial realization of simplified states $x_s^1, \ldots, x_s^{k_t}$ obtained from t-SNE. The safe and unsafe training data are denoted by green and blue points, respectively.

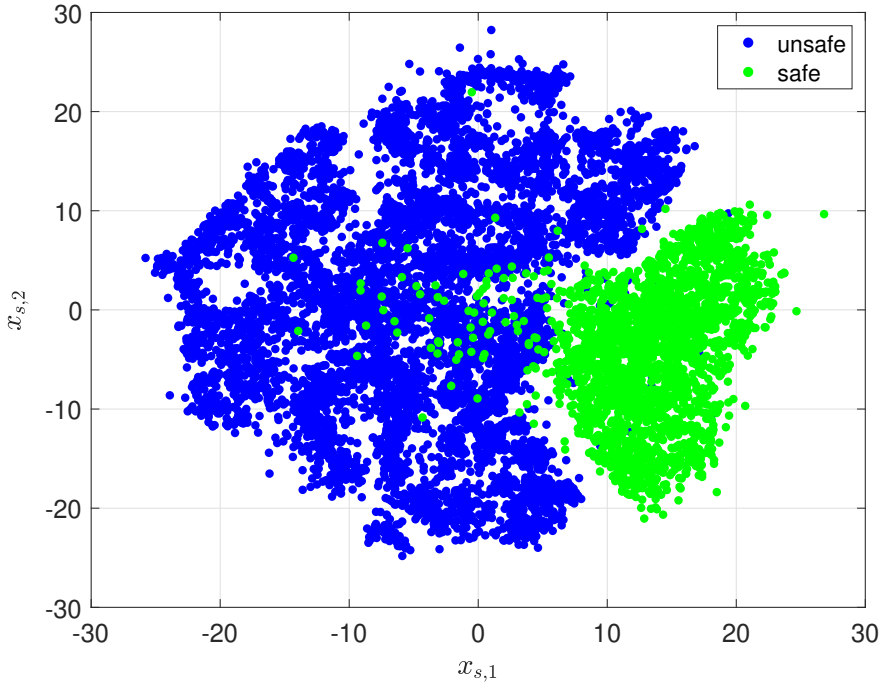


Figure 4.8.: The final realization of simplified states $x_s^1, \ldots, x_s^{k_t}$ obtained by recomputing with the learned neural network that represents the state mapping $x_s = \Psi(x) = \mathrm{NN}(x)$.

error, certain simplified states have a slightly changed position compared to the values obtained from t-SNE. However, this does not affect the computation of the low-dimensional representation of the safe region $\mathcal{S}_l$, as the results are updated later in the online adaptation using the feedback data.

We set the simplified state space as $\{\mathcal{X}_s \mid -30 \leq y_1, y_2 \leq 30\}$. By discretizing the simplified state space $\mathcal{X}_s$ into grid cells with step size 1 in both $x_{s,1}$ and $x_{s,2}$, we obtain the index vector $v \in \{1, 2, \ldots, 60\}^2$. The prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ is thus computed from the training dataset $\mathcal{D}_{\mathrm{train}}$ using the index vector v. The results are given in Fig. 4.9a, where the initial subjective uncertainty, the initial estimate and the minimum number are selected as $\sigma_{\mathrm{ini}} = 0.4$, $\mathbb{B}_{\mathrm{ini}} = (0.05, 0.55, 0.4)$ and $k_{\mathrm{min}} = 3$, respectively. Depending on the number of safe and unsafe training data in each grid cell, the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ estimates the probability $\mathbb{P}(x \in \mathcal{S})$ for original system states x that take the index vector v from the locating function $L(x)$. In Fig. 4.11a, the DSAF $\Gamma_{\mathrm{d}}(v)$ is initialized by the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$. In the next subsection, we demonstrate the update process of the DSAF $\Gamma_{\mathrm{d}}(v)$ using the proposed online adaptation method.

Updating the low-dimensional representation

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

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

The results of the online adaptation are given in Fig. 4.9-4.11. Prior to the update (update iteration $N = 0$), the DSAF $\Gamma_{\mathrm{d}}(v)$ is initialized as the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$, while the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ is constructed using the empty BBA $\mathbb{B}_{\varnothing}$ (see Fig. 4.9a, 4.10a and 4.11a). Once the learning procedure has started, we collect the feedback data incrementally. In the early updating phase, e.g., update iteration $N = 10$ with 200 feedback data, the DSAF $\Gamma_{\mathrm{d}}(v)$ is mainly determined by the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$. The subjective uncertainties of each training data are modified using the feedback data, where we become confident about the safety of certain training data when we observe a nearby feedback data that has the same safety property. Since the amount of feedback data is insufficient for providing a reliable safety estimate, the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ has a smaller effect on the computation of the low-dimensional representation of the safe region $\mathcal{S}_l$ (see Fig. 4.9b, 4.10b and 4.11b).

When more feedback data are available, e.g., update iteration $N = 50$, the feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ is able to provide more accurate safety estimates, hence its influence on the DSAF $\Gamma_{\mathrm{d}}(v)$ also becomes more significant. Due to the high dimensionality of the original system state x and the limited amount of feedback data, it is difficult to acquire an estimate with high confidence from the GPR model $\mathrm{GP}(x)$. As a result, changes are marginal in the prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ (see Fig. 4.9c, 4.10c and 4.11c). With even more feedback data, e.g., update iteration $N = 100$, the DSAF $\Gamma_{\mathrm{d}}(v)$ is able to provide reliable estimates about the

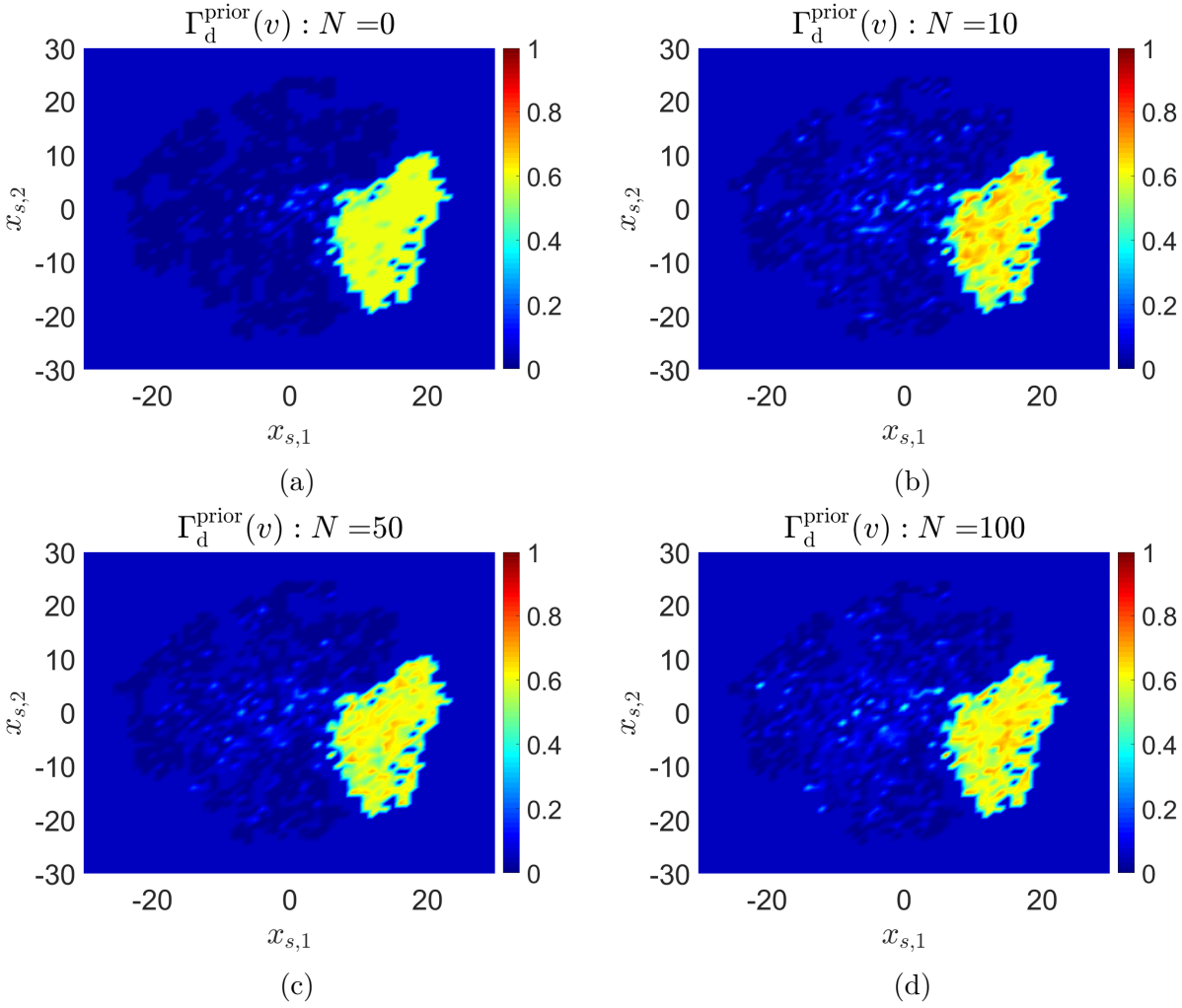


Figure 4.9.: The prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ in different update iterations N. $N = 0$ refers to the initialization prior to the online adaptation. The values of the safety estimates are represented by different colors. For example, blue and red colors represent unsafe and safe regions, respectively. While yellow color indicates the grid cells where no strong safety estimate can be made.

probability $\mathbb{P}(x \in \mathcal{S})$ for each index vector v. While the prior and feedback DSAFs are updated accordingly, the DSAF $\Gamma_{\mathrm{d}}(v)$ represents the actual low-dimensional representation of the safe region $\mathcal{S}_l$ under the unknown part of the system dynamics $d(x)$ (see Fig. 4.9d, 4.10d and 4.11d).

Comparison with physically inspired MOR

We compare the proposed approach with the physically inspired MOR in terms of the representation power of the identified low-dimensional representation of the safe region $\mathcal{S}_l$, i.e., how well the safe and unsafe states are separated. To do this, we compute another DSAF $\Gamma_{\mathrm{d}}(v)$ using physical features. As in Chapter 3, the low-dimensional safety feature, i.e., the simplified state x_s, is selected for the velocities in x and y directions $y = [v_x, v_y]^T$. To avoid any dangerous behavior in early learning phase, the low-dimensional representation of the

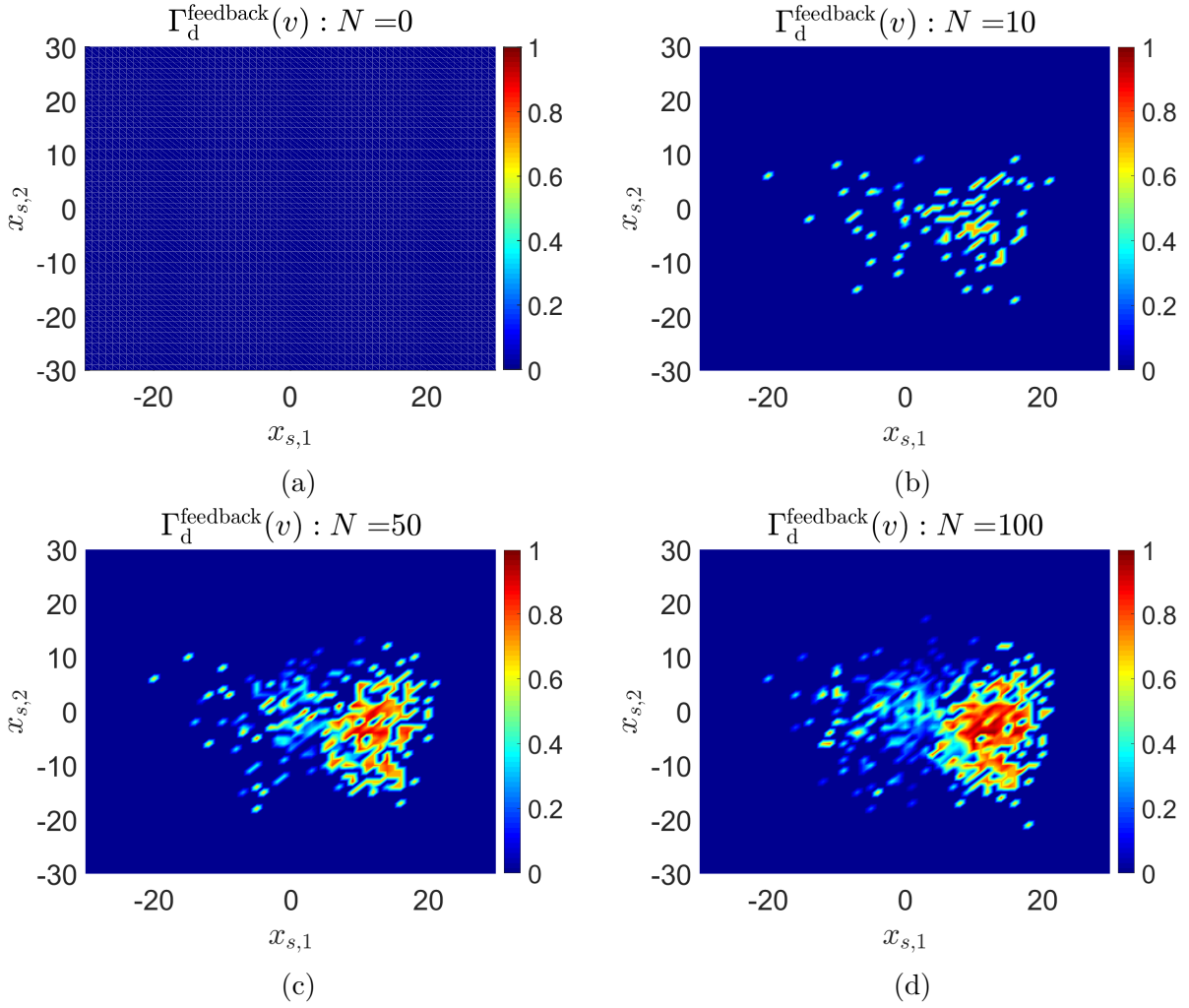


Figure 4.10.: The feedback DSAF $\Gamma_{\mathrm{d}}^{\mathrm{feedback}}(v)$ in different update iterations N. The values of the safety estimates are represented by different colors such as in Fig. 4.9. Grid cells with less observed feedback data appear to be small dots in the figures.

safe region $\mathcal{S}_l$ is initialized conservatively by setting $\Gamma_{\mathrm{d}}(v) = 0.6$ for grid cells that satisfy $-0.5 \leq v_x, v_y \leq 0.5$ (see Fig. 4.12).

If the learning task of the quadcopter is relatively simple, e.g. flying forward as given in Chapter 3, then a satisfying policy can be found without an extensive exploration in the state space. In such a case, the exploration is limited to a small subspace around the origin of the original system state space (see Section 4.5.1 for more discussions on this point). Therefore, to make a fair comparison, we also generate another feedback dataset $\mathcal{D}'_{\mathrm{feedback}}$ that has the same size as the dataset $\mathcal{D}_{\mathrm{feedback}}$. However, instead of using the complete original system state space, the states x_{real} in the set $\mathcal{D}'_{\mathrm{feedback}}$ are sampled from a smaller state space, where the ranges of angular positions and angular velocities are changed to $-\frac{1}{3}\pi \leq \theta_r, \theta_p, \theta_y \leq \frac{1}{3}\pi$ rad and -3 rad/s $\leq \omega_r, \omega_p, \omega_y \leq 3$ rad/s, respectively.

We first compare the performance of both approaches by considering a small state space, i.e., the feedback dataset $\mathcal{D}'_{\mathrm{feedback}}$ is used for the update. The results show that, in this case, physical features are able to provide reasonable predictions about safety, i.e., the safe and unsafe regions are separated (see Fig. 4.13a). Meanwhile, the proposed data-driven MOR

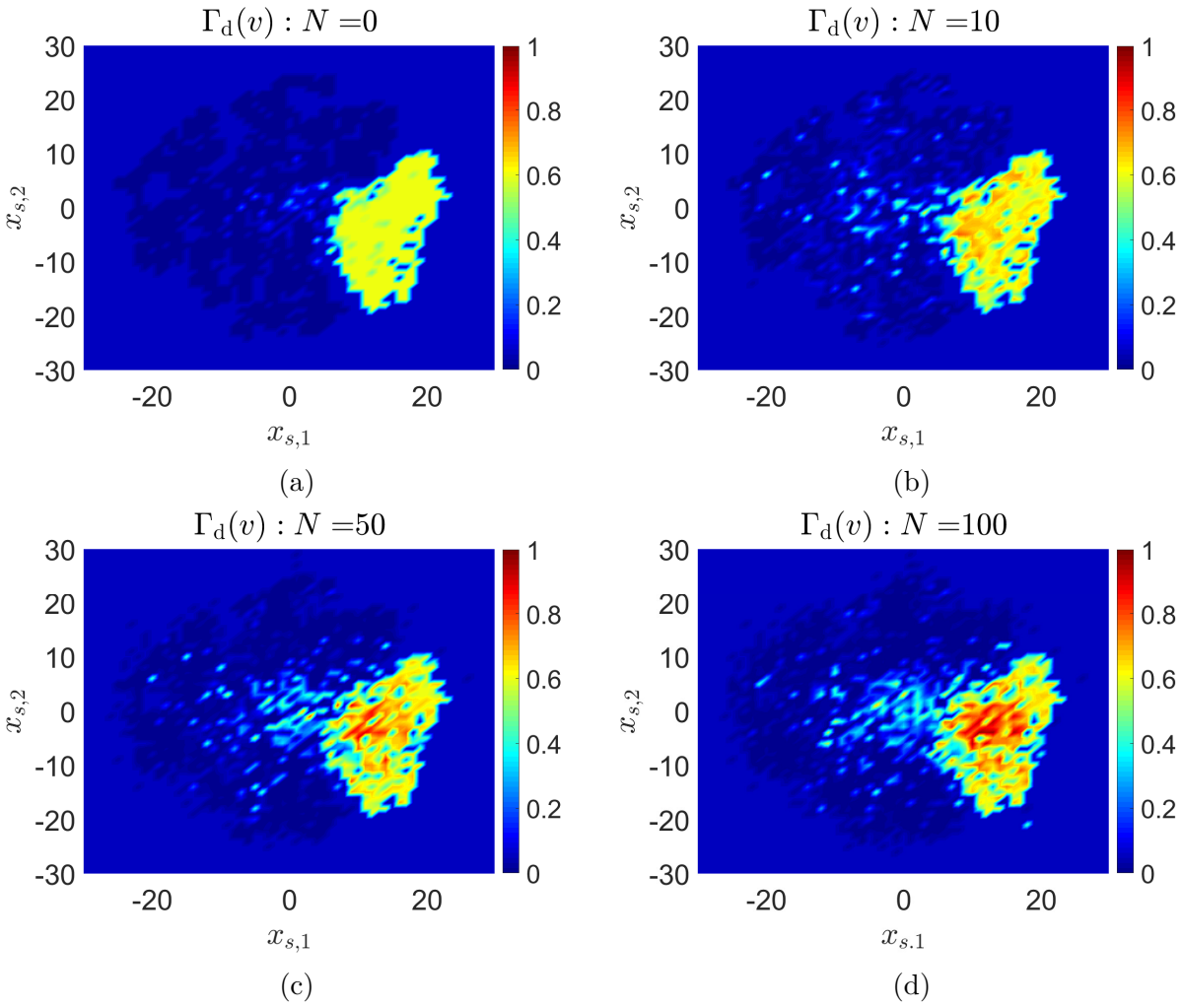


Figure 4.11.: The final obtained DSAF $\Gamma_d(v)$ in different update iterations N. The values of the safety estimates are represented by different colors such as in Fig. 4.9.

approach also produces a satisfying result with a marginally better separation between safe and unsafe states (see Fig. 4.13b).

However, if the learning task becomes more complex, the complete state space has to be explored to enable an optimal policy to be found. To simulate this scenario, we update the initial DSAF $\Gamma_d(v)$ using the feedback dataset $\mathcal{D}_{\text{feedback}}$. As seen in Fig. 4.14, when considering the entire original system state space, it is difficult to make reliable safety estimates based only on physical features. The boundary between safe and unsafe regions becomes unclear, and there are numerous grid cells that lead to a safety estimate close to 0.5. In contrast, the proposed approach is still able to find a representative low-dimensional representation of the safe region $\mathcal{S}_l$ for the complete state space. As the identified simplified state x_s can describe the safety of original system states x more precisely, a satisfying separation between safe and unsafe regions is achieved (see Fig. 4.11d) and more useful safety estimates are obtained. The independence of the size of the state space indicates the possibility of implementing the proposed approach on different learning tasks, which in turn increases the applicability of the SRL framework.

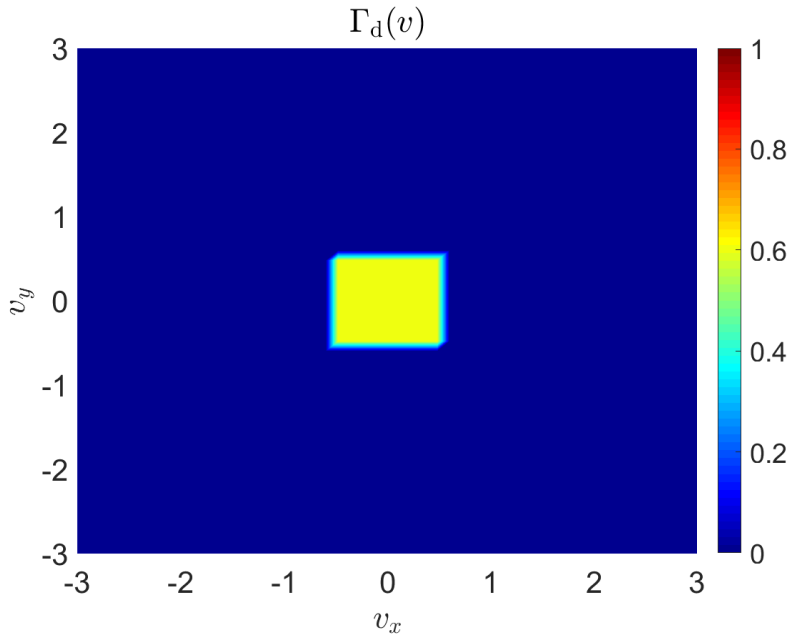


Figure 4.12.: For physically inspired MOR, the DSAF $\Gamma_{\mathrm{d}}(v)$ is initialized conservatively.

4.4.2. Humanoid Example

To further verify the performance of the proposed approach, an example of a humanoid robot is presented in this section. We employ the same Atlas humanoid model used in Section 3.5.3 and constrain it so that it is only able to move in the X-Z vertical plane (see Fig. 4.15). The 18-dimensional system state of the humanoid consists of the global positions and velocities of the body frame with respect to the ground frame, six joint angles and six joint velocities. The control input comprises the six motor torques applied to the hip, knee and ankle joints of both legs. We consider the safe region as the set of system states in which the humanoid can be controlled back to a safe standing posture without falling. The balance controller, i.e., the corrective controller $K(x)$, is a 1-step balance controller that allows the humanoid to take one step to balance itself with a predefined maximum step length. See literature on humanoid control, e.g., [37], for more details about this balance controller. We perform the simulation in Gazebo simulator with Open Dynamics Engine as the physics engine.

The training dataset $\mathcal{D}_{\text{train}}$ is generated by randomly sampling 10000 system states within the state space given by the DRCSIM[2]. We use the same parameters as in the quadcopter example for identifying the low-dimensional representation of the safe region of the humanoid. The initial low-dimensional representation of the safe region, i.e., the prior DSAF $\Gamma_{\mathrm{d}}^{\text{prior}}(v)$, is given in Fig. 4.16a. For simulating the reality gap, we consider the mass of the torso as 30% heavier. A feedback dataset $\mathcal{D}_{\text{feedback}}$ with 2000 randomly generated system states is then constructed for updating the low-dimensional representation of the safe region using the proposed online adaptation method. The updated low-dimensional representation of the safe region, i.e., the DSAF $\Gamma_{\mathrm{d}}(v)$, is presented in Fig. 4.16b. Similar results are observed to those obtained with the example of the quadcopter, which verifies both the performance and the applicability of the proposed approach to different learning tasks.

[2] `https://bitbucket.org/osrf/drcsim/`

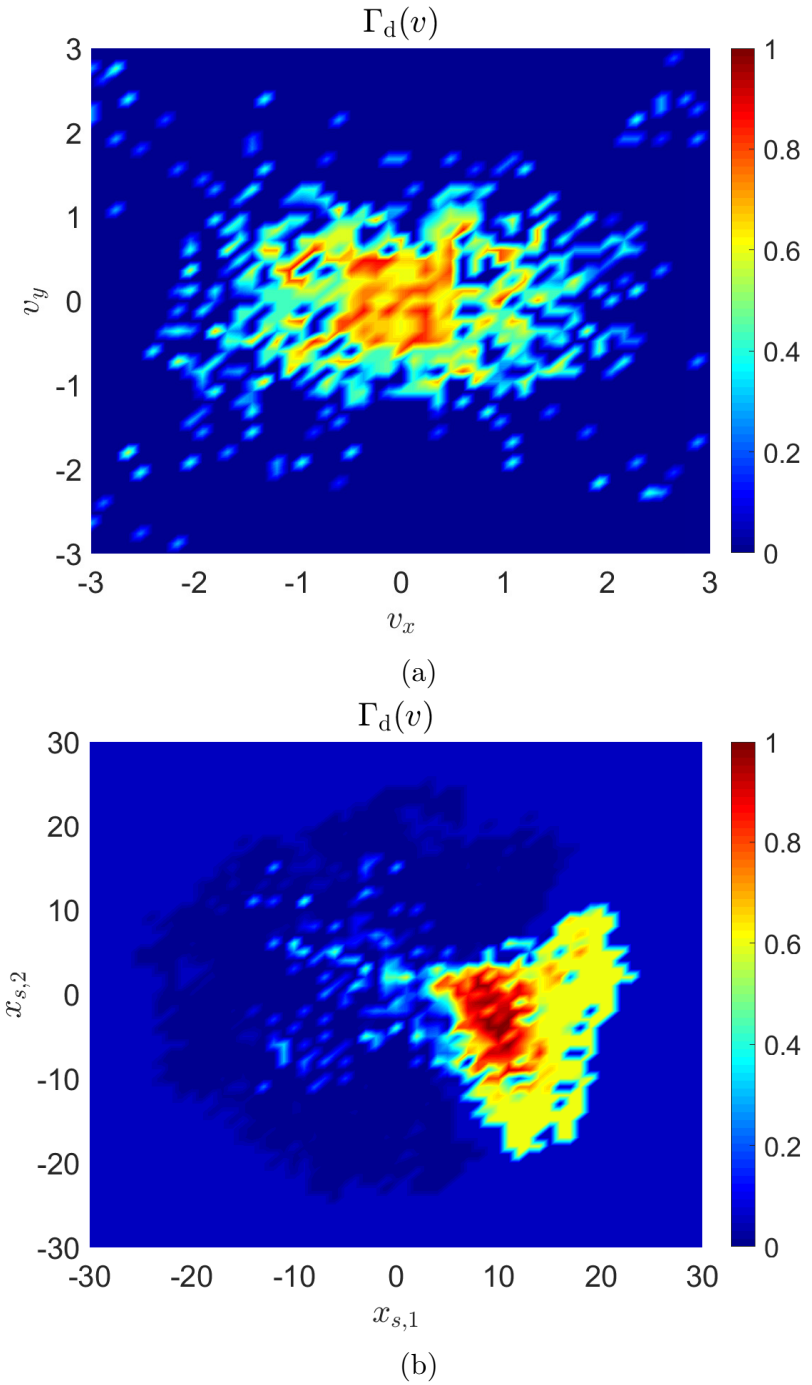


Figure 4.13.: The DSAFs $\Gamma_{\mathrm{d}}(v)$ obtained by using physically inspired MOR and the proposed approach, respectively. The feedback dataset $\mathcal{D}'_{\mathrm{feedback}}$ is used for the update. The values of the safety estimates are represented by different colors such as in Fig. 4.9.

4.5. Discussion

In this work, we propose a general data-driven approach to efficiently identify a low-dimensional representation of the safe region for the SRL framework. Two important aspects of the pro-

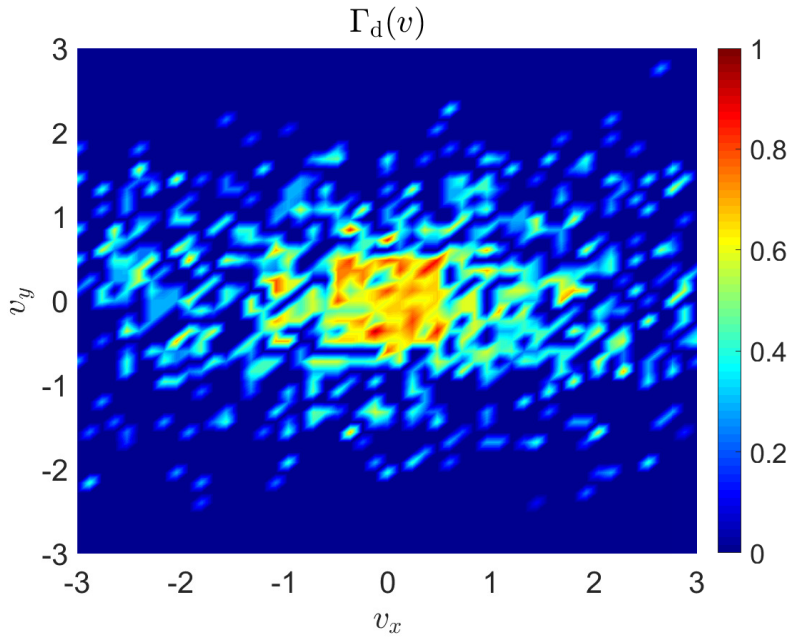


Figure 4.14.: The DSAF $\Gamma_{\mathrm{d}}(v)$ obtained by using physically inspired MOR and the feedback dataset $\mathcal{D}_{\text{feedback}}$. The values of the safety estimates are represented by different colors such as in Fig. 4.9.

posed approach are discussed in this section.

4.5.1. Relevance to Different SRL Tasks

In Chapter 3, the SRL framework utilizes a low-dimensional representation of the safe region $\mathcal{S}_l$ that is obtained using physically inspired MOR. Such a low-dimensional representation is useful when the learning task is relatively simple, e.g., teaching a quadcopter to fly forwards as given in Section 3.5.2, such that a satisfying control policy can be found without requiring an extensive exploration in the original state space. Since, in this case, the system state is likely to stay in a sub-state space near the origin, physical features are able to provide reliable safety estimates. However, when the learning task becomes more difficult, e.g., the quadcopter needs to track a complex 3D trajectory, the learning algorithm in general has to explore a large portion of the state space to find an optimal policy. Under these circumstances, at least a rough safety assessment of the complete state space is needed. Unfortunately, being restricted by the representation power, the physically inspired low-dimensional representation of the safe region $\mathcal{S}_l$ fails to provide useful safety estimates when considering the entire state space. Hence, the performance of the SRL framework is affected.

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

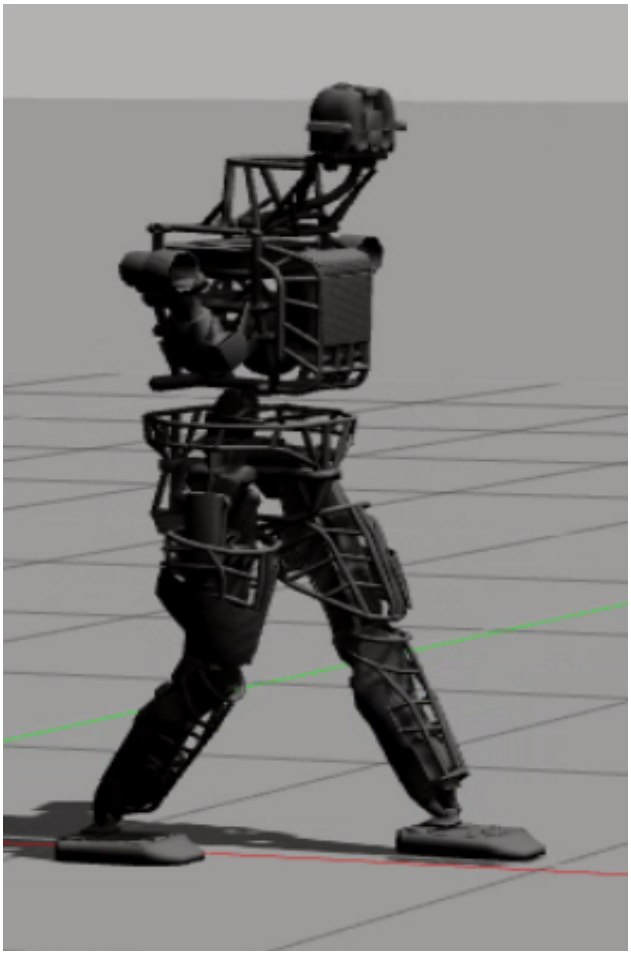
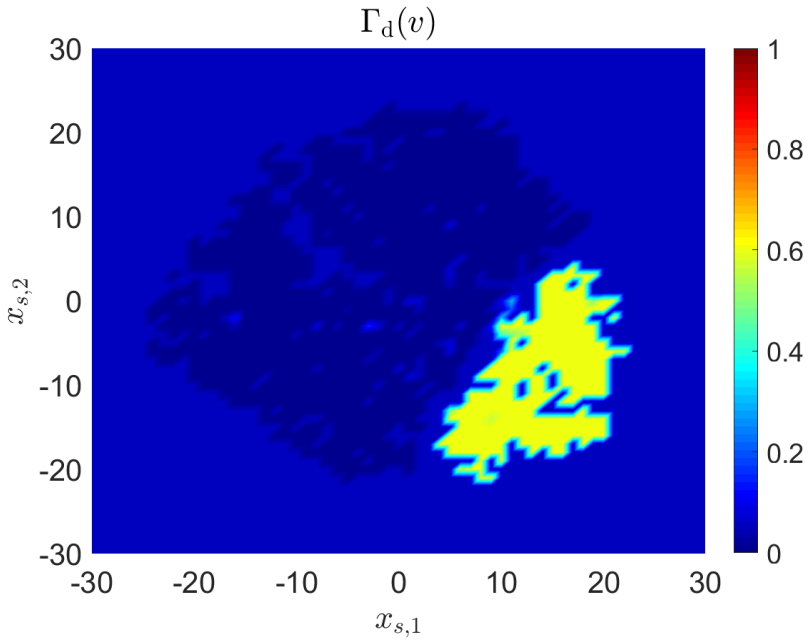


Figure 4.15.: Atlas humanoid model in the Gazebo simulation environment.

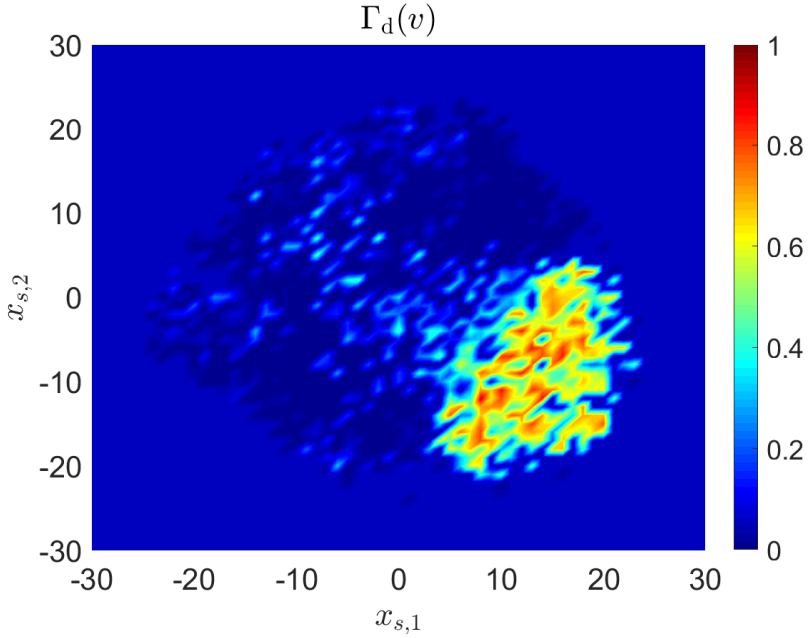
4.5.2. Strengths and Limitations

The presented approach has three particular strengths. First, it finds a low-dimensional representation of the safe region $\mathcal{S}_l$ that allows safe and unsafe states to be clearly separated for large portions of a high-dimensional state space; see also Section 4.4.1. Second, the effort required for identifying the low-dimensional representation of the safe region $\mathcal{S}_l$ is low. While, for instance, physically inspired MOR usually needs a comprehensive analysis of the system dynamics, the proposed approach relies solely on training data that can be collected efficiently even for complex dynamical systems through parallel computing and a suitable simulation environment. Third, it fully utilizes the information contained in the feedback data using two DSAFs. Hence, the update can be performed with few feedback data while providing a satisfying result.

However, the performance of the identified low-dimensional representation of the safe region $\mathcal{S}_l$ is affected by the quality of the nominal system, i.e. the magnitude of the discrepancy between the nominal and the real systems. While the state mapping $x_s = \Psi(x)$ is determined using only training data, the online adaptation method attempts to find an accurate DSAF $\Gamma_\mathrm{d}(v)$ based on the learned low-dimensional safety feature. If the reality gap is too large, then it is possible that the learned safety feature is not sufficiently representative and we might therefore observe more grid cells with final safety estimates that are close to 0.5,



(a)



(b)

Figure 4.16.: (a) The prior DSAF $\Gamma_{\mathrm{d}}^{\mathrm{prior}}(v)$ used as the initial low-dimensional representation of the safe region. The values of the safety estimates are represented by different colors such as in Fig. 4.9. (b) The updated DSAF $\Gamma_{\mathrm{d}}(v)$ for the low-dimensional representation of the safe region.

i.e., $\Gamma_{\mathrm{d}}(v) \approx 0.5$, which are less useful for guiding the learning process. In general, if the nominal system is assumed to be unreliable, a high probability threshold p_t should be used for constructing the low-dimensional representation of the safe region $\mathcal{S}_l$ (see (4.16)), such that the learning process becomes more conservative for keeping the system safe. However,

we usually consider the unknown system dynamics $d(x)$ as bounded within a reasonable range, since it makes less sense to use a dissimilar nominal system to predict the behavior of the real system. To further generalize the proposed approach, more studies are required to quantify the influence of the simulation-to-reality gap on the reliability of the obtained safety estimates.

4.6. Summary

To apply SRL to complex dynamical systems, we propose in this chapter a novel data-driven approach to identify a low-dimensional representation of the safe region for realizing a general SRL framework. Using a nominal system model that predicts the behavior of the real system, we first collect training data about the safety of different system states. Then, by computing the probabilistic similarities between each training data using a data-driven method called t-SNE, an initial realization of simplified states that best represents the training data is obtained. Such a realization determines the state mapping that maps each original system state to a corresponding simplified state. Afterwards, we generate an initial low-dimensional representation of the safe region according to the training data and the identified state mapping. To compensate for the mismatch between the nominal and the real systems, an efficient online adaptation method based on belief function theory is also proposed to update the low-dimensional representation of the safe region by accounting for the real system behavior. Each update iteration consists of three steps: first, the prior belief about the safe region obtained from the training data is modified by using the feedback data; second, the observed system behaviour is used to construct another belief about the safe region; third, the two beliefs are fused together to formulate a more accurate and reliable low-dimensional representation of the safe region. Two examples, one quadcopter and one humanoid, are presented to demonstrate the performance of the proposed approach. Experimental results show that, compared to the physically inspired MOR, a more reliable and representative low-dimensional representation of the safe region is found using the data-driven approach. However, our approach has one major limitation that its performance is affected by the magnitude of discrepancy between the nominal and real systems. If the reality gap is assumed to be large, then it is likely that a less meaningful low-dimensional representation of the safe region will be obtained.

For future work, we intend to combine the data-driven method with model-based MOR techniques, e.g., projection-based MOR, to find an approach that is more robust to the simulation-to-reality gap when identifying the low-dimensional representation of the safe region. This is inspired by recent studies where data-driven approaches are used to solve complicated equations [73]. While the major problem of employing projection-based MOR on complex dynamical systems is the computational difficulty in solving the relevant partial differential equations, data-driven approaches may provide a solution to this challenging problem. If a more clear mathematical connection between safety of the original and the simplified systems can be derived, then it is also possible to realize a more reliable safety guarantee for complex dynamical systems. Another possibility of future work is that, finding a way to quantify the similarity between different dynamical systems, such that the learned safety feature can be generalized from one system to other similar systems. If this can be achieved, then the training data can also be collected from multiple similar systems, which will highly increase the learning efficiency. However, how the similarity between dynamical

systems will be measured is still an open research problem.

Data Generation for Data-driven Safe Reinforcement Learning

5.

While the data-driven MOR enables the application of the SRL framework to a wider range of dynamical systems and learning scenarios, the performance is affected by the quality of the training data. This is due to the fact that, the low-dimensional safety feature and the state mapping are determined only by using the training data. If the identified safety feature is less representative, i.e., the safe and unsafe system states cannot be well separated in the simplified state space, then it is difficult to achieve meaningful safety estimates even if the proposed online adaptation method tries to modify the estimated safe region by accounting for the real system behavior.

Therefore, to ensure a satisfying performance of the data-driven SRL framework, especially in the early learning phase when only limited feedback data are available, we propose in this chapter a data generation method for producing representative training data. To do this, we first consider the estimation of safety of a given original system state as a binary classification problem. Then, we analyze how the classification error is affected by the training data as well as its relationship to the learning performance. Based on the results of this analysis, we design accordingly a data generation method that combines a uniform distribution and a multivariate normal distribution. While the uniform distribution preserves conservativeness during the learning, the multivariate normal distribution encourages explorations for a better policy. Hence, by adjusting the weights of these two distributions in the data generation process, a balance between finding an optimal control policy and keeping the system safe is achieved. The results show that, by using the proposed approach, we obtain a better initialization of the identified low-dimensional representation of the safe region, which leads to a better performance of the SRL framework.

The remainder of this chapter is organized as follows: we first explain how the examination of safety of original system states is considered as a binary classification problem in Section 5.1. Then, we discuss the relationship between learning performance and the classification error in Section 5.2. Thereafter, a data generation method, which is able to produce training data that are most useful to the SRL framework, is proposed in Section 5.3. A three-link inverted pendulum example is presented in Section 5.4 to demonstrate the performance of the proposed approach. Finally, a discussion and a summary are given in Section 5.5 and Section 5.6, respectively.

5.1. SRL as Binary Classification

The essential part of realizing a SRL approach is the determination of safety of a given original system state, i.e., whether the corrective controller is able to drive the system back to a safe state or not. In this section, we explain the relation between the examination of

safety and the binary classification.

We consider the real system as

$$\dot{x} = f(x) + g(x)u + d(x) \tag{5.1}$$

which is the same as given in (4.5). The safe region of the real system is denoted as $\mathcal{S}$. Then according to the SRL framework that is based on the supervisory control strategy, a labeling function $l(x) : \mathcal{X} \to \mathcal{Z} = \{1, 0\}$ categorizes the system states into safe and unsafe classes as

$$l(x) = z = \begin{cases} 1, & \text{if } x \in \mathcal{S} \backslash \{\partial \mathcal{S}\} \\ 0, & \text{else} \end{cases} \tag{5.2}$$

where safe and unsafe states have their safety labels as $z = 1$ and $z = 0$, respectively. Such a classification can be considered as a binary classification problem [122]. However, due to the computational difficulty under complex dynamicas, as well as the unknown and unmodelled part of system dynamics $d(x)$, it is infeasible to compute the safe region $\mathcal{S}$ and the ground-truth labeling function $l(x)$.

To solve this problem, a simplified system is introduced to compute a low-dimensional representation of the safe region $\mathcal{S}_l$. By mapping each system state x to a low-dimensional simplified state x_s with a state mapping $x_s = \Psi(x)$, the safety of system state x is estimated through safety of the corresponding simplified state x_s in a probabilistic form as

$$\mathbb{P}(l(x) = 1) = \Gamma(x_s)|_{x_s = \Psi(x)} \sim [0, 1] \tag{5.3}$$

where $\Gamma(x_s)$ is a SAF defined over the simplified state space $\mathcal{X}_s$ (see also Section 4.1.1). The low-dimensional representation of the safe region $\mathcal{S}_l$ is thus given by employing a probability threshold p_t as

$$\mathcal{S}_l = \{x_s \in \mathcal{X}_s \mid \Gamma(x_s) > p_t\} \tag{5.4}$$

which approximates the high-dimensional safe region $\mathcal{S}$.

A data-driven MOR approach can be used to find a reliable low-dimensional representation of the safe region $\mathcal{S}_l$. For this, we assume that a nominal system model is available as

$$\dot{x} = f(x) + g(x)u \tag{5.5}$$

By simulating the nominal system with a predefined corrective controller $K(x)$, we obtain a set of training data $\mathcal{D}_{\text{train}}$ that is defined as follows.

Definition 10. *The training dataset of k_t training data is given as*

$$\mathcal{D}_{\text{train}} = \{D^1_{\text{train}}, D^2_{\text{train}}, \ldots, D^{k_t}_{\text{train}}\}. \tag{5.6}$$

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

$$D^i_{\text{train}} = \{x^i, s(x^i), \Phi(t; x^i)\}. \tag{5.7}$$

x^i is the initial system state in which the corrective controller $K(x)$ is activated. $s(x^i)$ is the safety label that represents the result of safety recovery for the state x^i. We denote $s(x^i) = 1$ if the system state x^i is safe under the corrective controller $K(x)$, and $s(x^i) = 0$ if it is not. $\Phi(t; x^i)$ is the corresponding system trajectory of the safety recovery that starts at x^i when time $t = 0$.

Provided with the training dataset $\mathcal{D}_{\text{train}}$, data-driven approaches, e.g., t-SNE as given in Chapter 4, can be used to find a state mapping $x_s = \Psi(x)$ and a SAF $\Gamma(x_s)$ that best represent the training data. This results in an initial low-dimensional representation of the safe region $\mathcal{S}_l$. The data-driven SRL framework then utilizes the low-dimensional representation of the safe region $\mathcal{S}_l$ for estimating the safety of different real system states, which leads to the following hypothesis $h(x) : \mathcal{X} \to \mathcal{Z}$

$$h(x) = z = \begin{cases} 1, & \text{if } \Psi(x) = x_s \in \mathcal{S}_l \\ 0, & \text{if } \Psi(x) = x_s \notin \mathcal{S}_l \end{cases} \tag{5.8}$$

which works as an approximation of the ground-truth labeling function $l(x)$. The reliability of the hypothesis $h(x)$ depends, on the one hand, on the magnitude of discrepancy between the nominal and the real systems. On the other hand, the quality of training data also affects the performance of the hypothesis $h(x)$. In next section, we discuss how to acquire a well-performed training dataset $\mathcal{D}_{\text{train}}$ for the data-driven SRL framework by analyzing the influence of training data on the classification error.

5.2. Learning Performance and Classification Error

In this section, we first analyze factors that affect the classification error in the binary classification problem in SRL. Then, as the goal of SRL is to find a satisfying controller for completing the given control task, we also discuss the relationship between learning performance and the classification error. Details of the analysis are given as follows.

5.2.1. Training Data and Classification Error

By considering the estimation of safety of real system states as a binary classification problem, we utilize the classification error as the first criterion when generating the training dataset $\mathcal{D}_{\text{train}}$.

We assume that during the learning, all system states of the real system, in which the corrective controller $K(x)$ is activated, are drawn from an unknown distribution $\mathcal{N}$. Meanwhile, the distribution of the nominal system states contained in the training dataset $\mathcal{D}_{\text{train}}$ is denoted as $\mathcal{N}_n$. Then for the learned hypothesis $h(x)$, its classification error on the real system $\epsilon(h, l)$ (referred to as the generalization error [123]) and on the nominal system $\epsilon_n(h, l_n)$ (referred to as the source error [123]) are

$$\epsilon(h, l) = \mathrm{E}_{x \sim \mathcal{N}} \left[\mathbb{I}(h(x) \neq l(x)) \right] \tag{5.9}$$

$$\epsilon_n(h, l_n) = \mathrm{E}_{x \sim \mathcal{N}_n} \left[\mathbb{I}(h(x) \neq l_n(x)) \right] \tag{5.10}$$

where $l_n(x)$ is the ground-truth labeling function that corresponds to the actual safe region of the nominal system. The generalization error and the source error represent the probability that according to the distribution $\mathcal{N}$ or $\mathcal{N}_n$, the hypothesis $h(x)$ disagrees with the ground-truth labeling function $l(x)$ given by the real system, or the ground-truth labeling function $l_n(x)$ given by the nominal system, respectively. By extending Theorem 1 given in [123] based on the $\mathcal{H}\Delta\mathcal{H}$-divergence, the following theorem that bounds the generalization error holds for the hypothesis $h(x)$.

Theorem 1. *The generalization error $\epsilon(h, l)$ of hypothesis $h(x)$ satisfies*

$$\begin{aligned}\epsilon(h,l) \leq \quad & \epsilon_n(h,l_n) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{N},\mathcal{N}_n) \\ & + \min\{\mathrm{E}_{x\sim\mathcal{N}_n}\left[\mathbb{I}(l(x)\neq l_n(x))\right], \mathrm{E}_{x\sim\mathcal{N}}\left[\mathbb{I}(l(x)\neq l_n(x))\right]\}\end{aligned} \tag{5.11}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ is the $\mathcal{H}\Delta\mathcal{H}$-distance.

Proof. Let $\epsilon_n(h,l) = \mathrm{E}_{x\sim\mathcal{N}_n}\left[\mathbb{I}(h(x)\neq l(x))\right]$, we have

$$\begin{aligned}\epsilon(h,l) &= \epsilon(h,l) + \epsilon_n(h,l_n) - \epsilon_n(h,l_n) + \epsilon_n(h,l) - \epsilon_n(h,l) \\ &\leq \epsilon_n(h,l_n) + |\epsilon_n(h,l) - \epsilon_n(h,l_n)| + |\epsilon(h,l) - \epsilon_n(h,l)| \\ &\leq \epsilon_n(h,l_n) + \mathrm{E}_{x\sim\mathcal{N}_n}\left[\mathbb{I}(l(x)\neq l_n(x))\right] + |\epsilon(h,l) - \epsilon_n(h,l)|\end{aligned} \tag{5.12}$$

According to Lemma 3 in [123], the following holds for any two hypothesis h_1 and h_2

$$|\epsilon(h_1,h_2) - \epsilon_n(h_1,h_2)| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{N},\mathcal{N}_n) \tag{5.13}$$

By considering the labeling function $l(x)$ as a hypothesis, (5.12) becomes

$$\epsilon(h,l) \leq \epsilon_n(h,l_n) + \mathrm{E}_{x\sim\mathcal{N}_n}\left[\mathbb{I}(l(x)\neq l_n(x))\right] + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{N},\mathcal{N}_n) \tag{5.14}$$

If in the first line we use $\epsilon(h,l_n) = \mathrm{E}_{x\sim\mathcal{N}}\left[\mathbb{I}(h(x)\neq l_n(x))\right]$ instead of $\epsilon_n(h,l)$, we have

$$\epsilon(h,l) \leq \epsilon_n(h,l_n) + \mathrm{E}_{x\sim\mathcal{N}}\left[\mathbb{I}(l(x)\neq l_n(x))\right] + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{N},\mathcal{N}_n) \tag{5.15}$$

Combining (5.14) and (5.15) hence gives the Theorem 1.

□

The upper bound of the generalization error given in (5.11) contains three terms: the first term is the source error; the second term represents the divergence in distributions; the third term is the difference in labeling functions and cannot be changed, as it is affected only by the discrepancy between the nominal and the real systems. Hence for achieving a low generalization error, Theorem 1 suggests to move closer the two distributions $\mathcal{N}$ and $\mathcal{N}_n$ while keeping the source error small. Based on this, it is motivated to generate the training dataset $\mathcal{D}_{\text{train}}$ by using an accurate estimate of the unknown distribution $\mathcal{N}$.

In Chapter 4, the training dataset $\mathcal{D}_{\text{train}}$ is generated by sampling system states with a uniform distribution $\mathcal{N}_{\text{ud}}$ among the entire state space. However when controlling a dynamical system, the probability that a system state x will be visited is affected by the system dynamics (see Section 5.4.2 and in particular Fig. 5.2a for an example). Hence, in general the UD $\mathcal{N}_{\text{ud}}$ is expected not to be close to the real distribution $\mathcal{N}$.

Although the distribution $\mathcal{N}$ is unknown prior to the learning process on the real system, it can be approximated by simulating the nominal system. To do this, we first set the initial state of the nominal system as the origin. Then, we control the nominal system with a random policy and record all system states observed in the system trajectory. Repeating this multiple times results in a dataset of system states that reflects the probability that different states will be visited during control. A multivariate normal distribution $\mathcal{N}_{\text{mnd}}(\mu,\Sigma)$ is then fitted to the collected dataset of system states and is considered as an approximation of the real distribution $\mathcal{N}$. Apparently, the accuracy of such an approximation is affected by the magnitude of discrepancy between the nominal and the real systems. Nevertheless, the training dataset $\mathcal{D}_{\text{train}}$ drawn from the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ becomes closer to the distribution $\mathcal{N}$.

5.2.2. Classification Error and SRL

The motivation of using the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ for generating training data points is from the prospective of reducing the generalization error. However, if the data generation is decided only based on the generalization error, the performance of the SRL framework might be affected due to the following reason.

The generalization error consists of two parts

$$\epsilon(h, l) = \mathrm{E}_{x\sim\mathcal{N}}\left[\mathbb{I}(h(x) = 1, l(x) = 0)\right] + \mathrm{E}_{x\sim\mathcal{N}}\left[\mathbb{I}(h(x) = 0, l(x) = 1)\right] \tag{5.16}$$

While the first error type (false positive) will cause unsafe behaviours of the dynamical system, the second error type (false negative) only means conservativeness in the SRL process. The purpose of SRL is to find a satisfying reinforcement learning-based policy $\pi(x)$ while keeping a high probability that the system is safe. Hence, conservativeness is acceptable as long as a well-performed policy can be learned within the subregion of state space restricted by the supervisor. In that regard, considering only the generalization error in the training data generation is likely to deteriorate the performance of the SRL framework, as reducing the conservativeness usually also means a higher chance of encountering an unsafe system behavior. This is also the reason why the probability threshold p_t of the SRL framework is usually chosen to be a high value, e.g., 0.8, instead of 0.5, which in the ideal case is the optimal value that minimizes the classification error according to the Bayes classifier [124].

Therefore, for taking the performance of the SRL framework into consideration, we in general would like to keep a certain degree of conservativeness during the learning process. This can be achieved by reducing our confidence in considering a system state x as safe unless enough evidence is provided. In that sense, using the uniform distribution $\mathcal{N}_{\text{ud}}$ for generating the training dataset $\mathcal{D}_{\text{train}}$ is helpful. The reason is that, compared to the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ that has a majority of data points being close to the origin, the training data points are now placed among the entire state space (see Section 5.4.2 for an example). Thus the proportion of safe data points is reduced. Note that, the underlying principle of using data-driven method for making safety predictions is to use known data points for estimating the safety of unseen data points. Therefore, the hypothesis $h(x)$ learned from the uniform distribution $\mathcal{N}_{\text{ud}}$ tends to make an unsafe prediction, since it is less likely to find a nearby safe training data point. As a result, although the uniform distribution $\mathcal{N}_{\text{ud}}$ gives a higher generalization error, it preserves the conservativeness in the SRL framework, which then ensures a higher probability that the system is safe during the learning process. For simultaneously considering the classification error and the learning performance, we propose in next section a data generation method that combines the uniform distribution $\mathcal{N}_{\text{ud}}$ and the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$.

5.3. Data Generation Method

While using the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ for generating the training dataset $\mathcal{D}_{\text{train}}$ reduces the conservativeness of the SRL framework, it also increases the risk of encountering an unsafe behavior. In contrast, the uniform distribution $\mathcal{N}_{\text{ud}}$ keeps the conservativeness and limits the learning process. However, if the reinforcement learning algorithm is overly restricted, it might not be able to find a satisfying policy for accomplishing the given

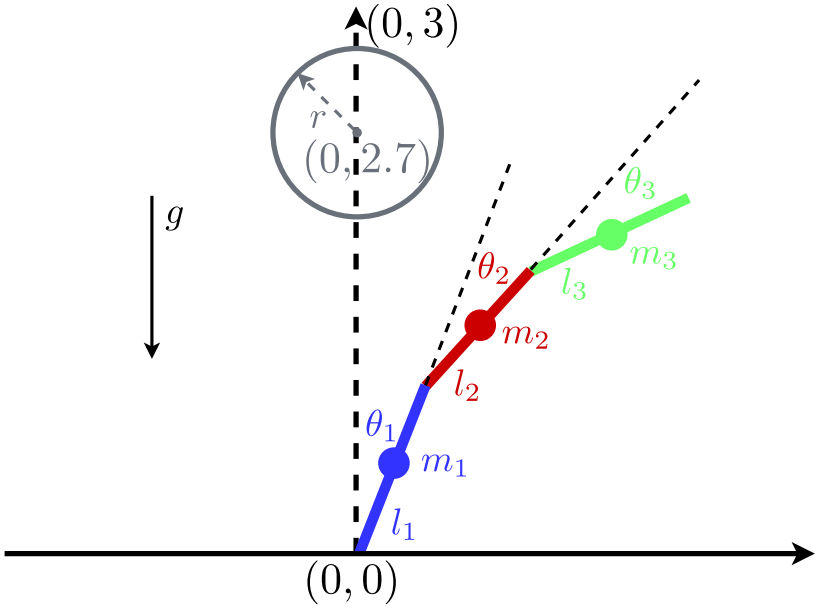


Figure 5.1.: Three-link inverted pendulum with a target circle given in the Cartesian space. The connection point between the pendulum and the ground is the origin of the Cartesian coordinate system. When the pendulum is at the zero configuration, the end-effector point locates at $(0, 3)$. The target circle has its centre at $(0, 2.7)$ and its radius as $r = 0.3$.

control task. Hence for achieving a good balance between the learning performance and the probability of being safe, we propose to divide the training dataset $\mathcal{D}_{\text{train}}$ into two parts

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{ud}} + \mathcal{D}_{\text{mnd}} \tag{5.17}$$

with $|\mathcal{D}_{\text{ud}}| = \lambda k_t$, $|\mathcal{D}_{\text{mnd}}| = (1 - \lambda)k_t$ and $0 \leq \lambda \leq 1$, where k_t is the size of the complete training dataset $\mathcal{D}_{\text{train}}$. The sub-datasets $\mathcal{D}_{\text{ud}}$ and $\mathcal{D}_{\text{mnd}}$ are generated by using the uniform distribution $\mathcal{N}_{\text{ud}}$ and the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$, respectively.

The coefficient λ determines the size of sub-datasets as well as the tendency of the SRL framework to perform exploration or to keep the safety. If the unknown dynamics $d(x)$ is assumed to be small, or failure of the corrective controller $K(x)$ is considered as less critical, it is suggested to use a small value of λ for ensuring a satisfying performance of the reinforcement learning algorithm, where training data points are mostly sampled by using known knowledge about the system trajectories. On the contrary, if it is more important to avoid unsafe behaviours, then a large value of λ should be used to keep the conservativeness in learning, i.e., most training data points are drawn from the uniform distribution $\mathcal{N}_{\text{ud}}$.

It is also worth mentioning that, while the probability threshold p_t stands for the minimal performance expectation of the SRL framework, the proposed data generation method provides more flexibility in balancing between learning and keeping the safety. It can work together with a high value of the probability threshold p_t for having a safer SRL framework, especially when the discrepancy between the nominal and the real systems is assumed to be large.

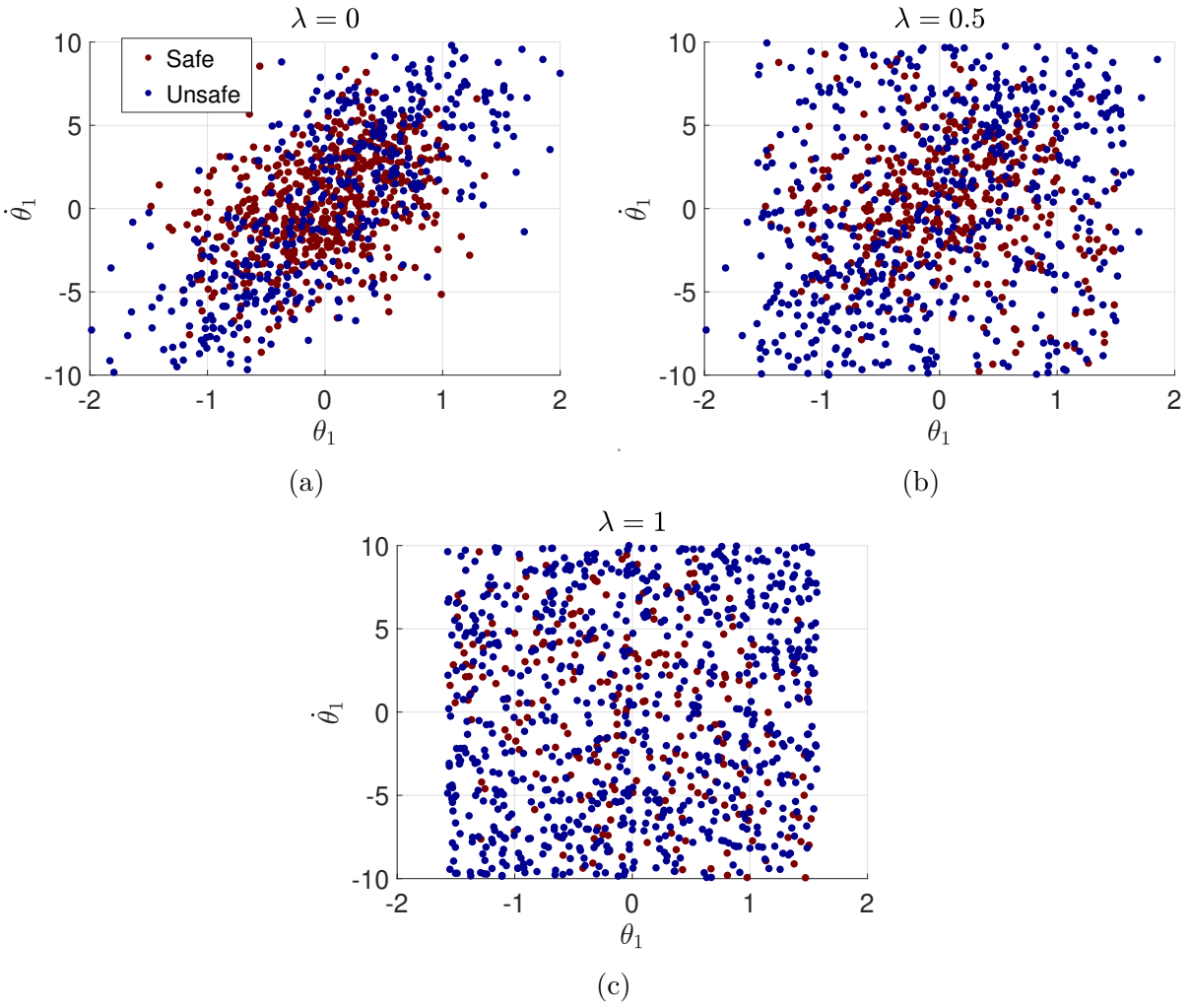


Figure 5.2.: Distribution of joint angle and angular velocity of the first link in the sampled system states for $\lambda = 0$, $\lambda = 0.5$ and $\lambda = 1$, respectively.

5.4. Experimental Results

In this section, we examine the influence of the proposed data generation method on the performance of the SRL framework with a three-link inverted pendulum example.

5.4.1. Experimental Setup

We consider a three-link inverted pendulum given as in Fig. 5.1. The system state is 6-dimensional that consists of three joint angles and three joint angular velocities as $x = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$. The origin of the system state space is the upright equilibrium point. The inputs $u = [u_1, u_2, u_3]^T$ are the torques applied on the three joints, where the maximal and minimal allowed torques are $u_{\max} = 100$ N m and $u_{\min} = -100$ N m for all three joints. The lengths of the links are set to $l_1 = l_2 = l_3 = 1$ m. We assume that the masses are concentrated on the CoMs that are located at the middle point of each link. For the nominal system, we consider the masses as $m_1 = m_2 = m_3 = 1$ kg. The discrepancy between the nominal and the real systems is assumed to be caused by the mismatch in the masses

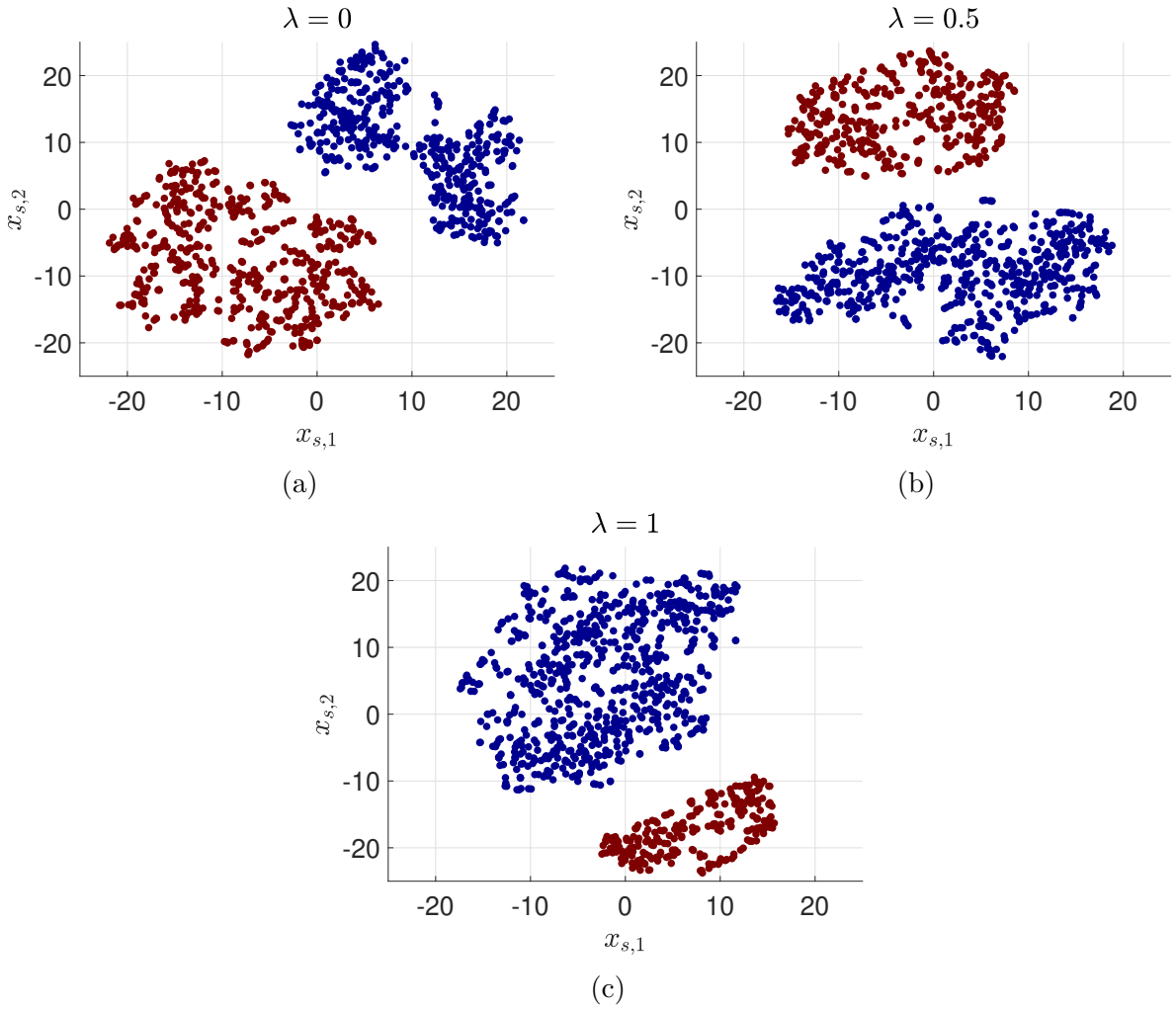


Figure 5.3.: The realization of simplified states computed by using t-SNE for $\lambda = 0$, $\lambda = 0.5$ and $\lambda = 1$, respectively.

of the first and the second links as $m_1 = m_2 = \Delta \cdot 1$ kg. The sample ranges of each state variable used in the uniform distribution $\mathcal{N}_{\text{ud}}$ are chosen as: $-\frac{\pi}{2}\text{rad} < \theta_1 < \frac{\pi}{2}\text{rad}$, $-\pi\text{rad} \leq \theta_2, \theta_3 \leq \pi\text{rad}$, $-10\text{rad/sec} \leq \dot{\theta_1} \leq 10\text{rad/sec}$, $-20\text{rad/sec} \leq \dot{\theta_2}, \dot{\theta_3} \leq 20\text{rad/sec}$.

The learning task is to find a control policy $\pi(x)$ that makes the end-effector point of the pendulum track a trajectory given as a circle in the Cartesian space with an angular velocity with respect to the centre of the circle as π rad/sec (see Fig. 5.1). We consider the connection point between the pendulum and the ground as the origin $(0, 0)$ of the Cartesian coordinate system. In each learning trial, the pendulum starts with the zero configuration $x = \mathbf{0}$, where the end-effector point is located at $(0, 3)$. The target circle has its centre located at $(0, 2.7)$ and its radius as $r = 0.3$. During the learning process, we attempt to keep the system safe by preventing the first link from hitting the ground.

We use the PPO algorithm [112] as the learning-based controller. The following reward function is used

$$R = R_c - 10 \cdot ||p_e - p_d|| \tag{5.18}$$

where $R_c = 2$ is a constant reward for being safe, $||p_e - p_d||$ is the Euclidean distance between

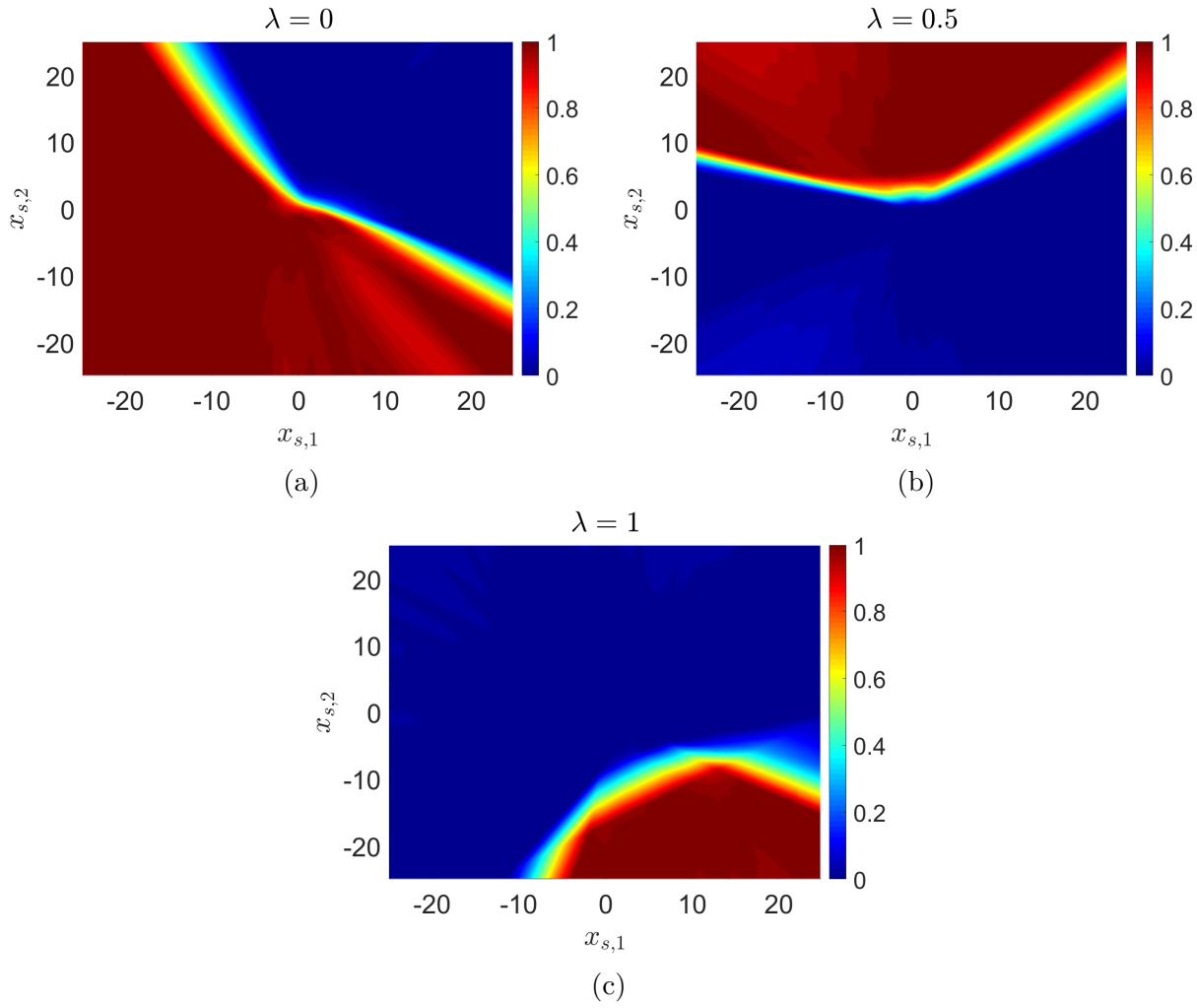


Figure 5.4.: The learned low-dimensional representation of the safe region $\mathcal{S}_l$. The output of the SAF $\Gamma(x_s)$ is represented by different colors.

the current end-effector position p_e and the desired position p_d given by the trajectory of the target circle. Following parameters are used for the PPO: the number of steps per update is 2048, the value function loss coefficient is 1, the gradient norm clipping coefficient is 10, the learning rate is $1e^{-4}$, the number of training epochs per update is 10, the number of training minibatches per update is 128, the discounting factor is 0.99, the advantage estimation discounting factor is 0.95, the policy entropy coefficient is 0, the number of hidden layers of the neural network is 2 with 128 neurons in each layer.

The corrective controller $K(x)$ is a LQG controller that is derived from the linearized nominal system model. When activated, the corrective controller $K(x)$ tries to control the system back to the upright configuration. For the SRL framework, the probability threshold is set to $p_t = 0.8$. Each learning condition is trained with three different seeds, and the averaged results are presented in the following subsections.

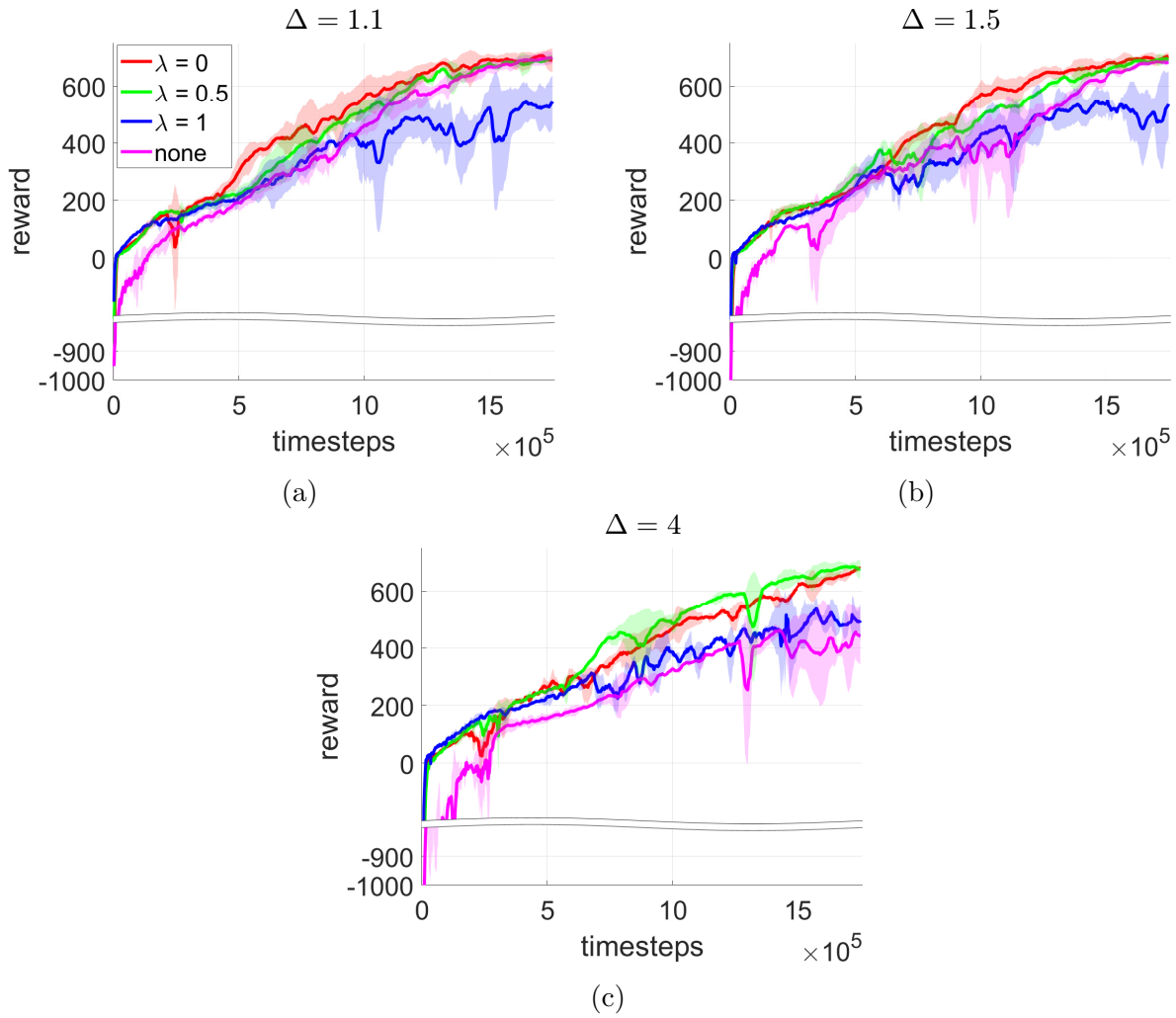


Figure 5.5.: Learning performance of the SRL framework with different low-dimensional representations of the safe region obtained with $\lambda = 0$, $\lambda = 0.5$, $\lambda = 1$ as well as the case that no supervisor is implemented (none), for $\Delta = 1.1$, $\Delta = 1.5$ and $\Delta = 4$, respectively.

5.4.2. Low-dimensional Representation of the Safe Region

We first examine the influence of the parameter λ on the learned low-dimensioanl representation of the safe region $\mathcal{S}_l$. We consider the training dataset $\mathcal{D}_{\text{train}}$ with size $k_t = 1000$ and generate it with three different values of λ: $\lambda = 0$ (using only the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$), $\lambda = 0.5$, and $\lambda = 1$ (using only the uniform distribution $\mathcal{N}_{\text{ud}}$). The corresponding results are presented in Fig. 5.2-5.4.

Fig. 5.2 shows the distributions of sampled system states contained in the generated training dataset $\mathcal{D}_{\text{train}}$ by displaying the joint angle and angular velocity of the first link. As the learning starts at the origin, it is more likely to observe a system state in the neighbourhood of the origin. Hence, the system states generated from the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ are more dense in the subregions near the origin (see Fig. 5.2a). As a result, the proportion of safe data points increases, since the closer to the origin, the higher the probability that a system state can be controlled back to the upright position. Moreover, limited by the

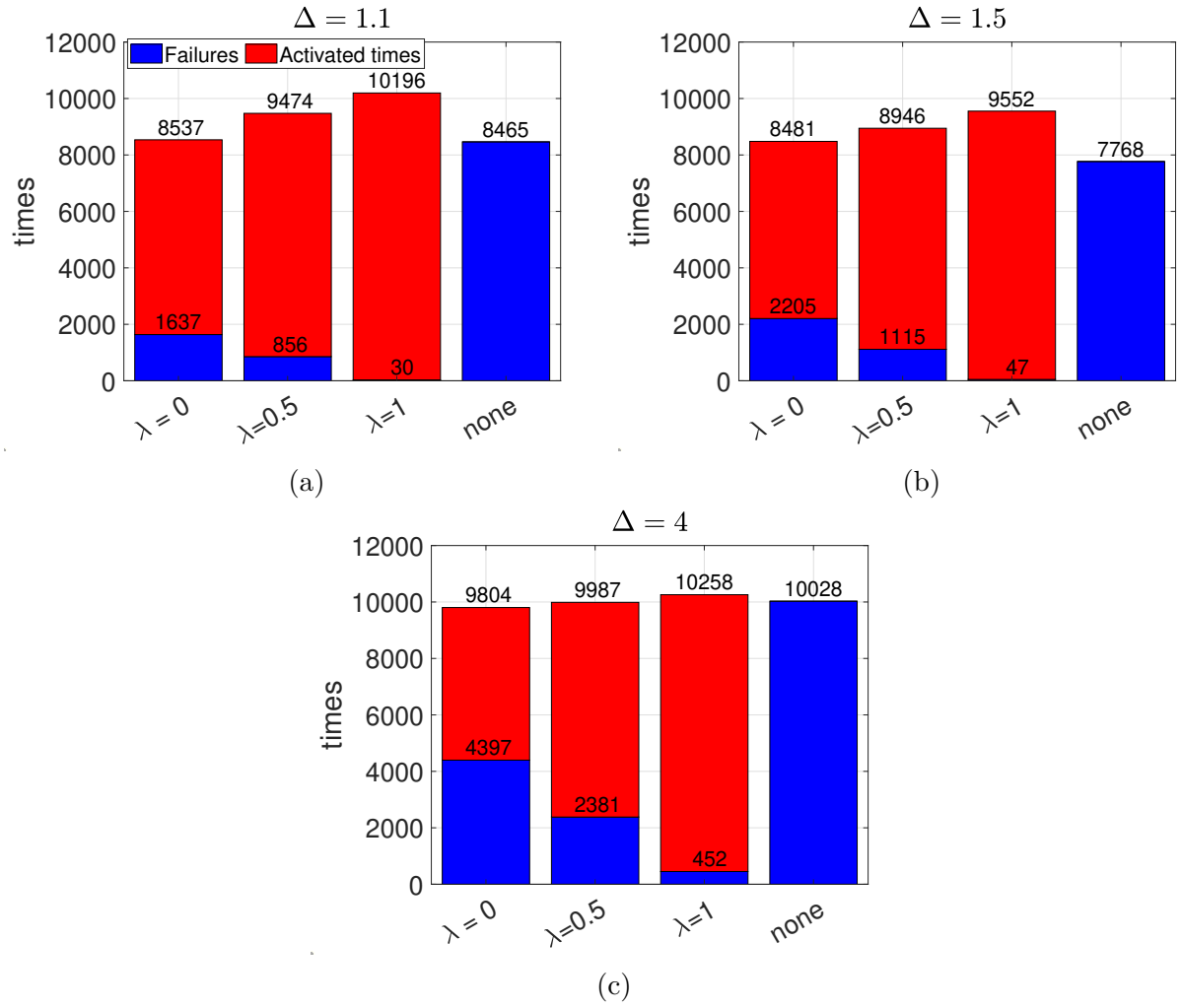


Figure 5.6.: The total number of times that the corrective controller $K(x)$ is activated and the corresponding number of failures for $\Delta = 1.1$, $\Delta = 1.5$ and $\Delta = 4$, respectively.

natural dynamics, there exists no feasible control sequence to control the system to a state that simultaneously has a large positive angle and a large negative angular velocity (right-bottom of Fig. 5.2a), or a large negative angle and a large positive angular velocity (left-top of Fig. 5.2a) of the first link. Therefore, no system states are sampled in those subregions. For the uniform distribution $\mathcal{N}_{\text{ud}}$, the sampled system states are placed among the entire state space, and as a consequence, a smaller proportion of safe data points is obtained (see Fig. 5.2c).

Fig. 5.3 gives the realizations of simplified states $\{x_s^1, \ldots, x_s^{kt}\}$ derived by using t-SNE. The safe and unsafe training data points are clearly separated in the simplified state space $\mathcal{X}_s$. The corresponding learned low-dimensional representation of the safe region $\mathcal{S}_l$ are presented in Fig. 5.4. With less observed safe training data points, the initial estimate of the low-dimensional representation of the safe region $\mathcal{S}_l$ becomes more conservative and tends to make an unsafe prediction. Note that, the axes $x_{s,1}$ and $x_{s,2}$ obtained with different training datasets have different meanings and cannot be directly compared, as they are the outputs of different t-SNE computations.

5.4.3. Performance of the SRL Framework

We then examine the influence of the low-dimensional representation of the safe region $\mathcal{S}_l$ learned from different training datasets $\mathcal{D}_{\text{train}}$ on the performance of the SRL framework. We compare the performance with three different levels of discrepancy between the nominal and the real systems as $\Delta = 1.1$, $\Delta = 1.5$ and $\Delta = 4$. Note that, since in this chapter we focus only on the training data generation, we use the initially learned hypothesis $h(x)$ throughout the entire learning process, such that its influence is better illustrated. As a baseline for comparisons, we also investigate the learning performance of implementing the PPO directly without the supervisor. The corresponding results are shown in Fig. 5.5 and Fig. 5.6.

As illustrated in Fig. 5.5, for all three levels of discrepancy, the training dataset that uses only the uniform distribution $\mathcal{N}_{\text{ud}}$, i.e., $\lambda = 1$, results in a final policy with a lower reward, since the learning process is overly restricted. For training dataset that uses only the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ ($\lambda = 0$) or that combines the uniform distribution $\mathcal{N}_{\text{ud}}$ and the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ ($\lambda = 0.5$), the SRL framework is able to find a satisfying final policy. It is also worth noting that, as each learning trial is terminated when an unsafe behaviour is predicted to occur, an early-stopping functionality is introduced to the learning process by using the SRL framework, which then helps with searching for an optimal policy. This effect becomes more significant when the system is hard to control due to heavier masses ($\Delta = 4$), where compared to the free learning case, a better final policy is found when using the SRL framework.

Fig. 5.6 shows the total number of times that the corrective controller $K(x)$ is activated during the entire learning process as well as the corresponding number of failures among these safety recoveries. As a comparison, the total number of times that the safety constraint is violated during the free learning case is also given. When the discrepancy is small, i.e., $\Delta = 1.1$, the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ results in a success rate of the corrective controller $K(x)$ (80.8%) that is close to the probability threshold $p_t = 0.8$. While increasing the value of λ makes the learning process more conservative, it also ensures a higher probability that the system is safe, e.g. 91.0% for $\lambda = 0.5$ and 99.7% for $\lambda = 1$. Similar behaviours can also be observed for $\Delta = 1.5$ and $\Delta = 4$, though the success rate of the corrective controller $K(x)$ decreases according to the level of the discrepancy. In general, the low-dimensional representation of the safe region $\mathcal{S}_l$ learned from the uniform distribution $\mathcal{N}_{\text{ud}}$ is more robust to the mismatches due to its higher conservativeness. However, if the discrepancy is assumed to be large, then an online adaptation method such as the one given in Chapter 4, needs to be introduced for ensuring a satisfying performance of the SRL framework.

5.5. Discussion

In this chapter, we introduce a data generation method for producing representative training data for the data-driven SRL framework. Two important aspects of the proposed approach are discussed in this section.

5.5.1. Connection to Transfer Learning

The proposed data generation method utilizes a multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ to estimate the unknown distribution of real system states $\mathcal{N}$ during the learning process. This is achieved by first sampling states from the nominal system with a random control policy, and then fitting the multivariate normal distribution $\mathcal{N}_{\text{mnd}}$ into the corresponding collected dataset. Such a process is in a way similar to the idea of transfer learning, where a control policy is first trained in simulation and then transferred to the real system. Hence, it is possible to connect the data generation method to the transfer learning method. For example, a reinforcement learning method can be first applied on the nominal system to find a satisfying control policy. And in the meantime, instead of using a random policy, the visited states during this pre-training process can be recorded for learning a more accurate estimate of the distribution of real system states $\mathcal{N}$. Later, when transfer learning attempts to transfer the learned policy from the nominal system to the real system, such a better estimate of distribution $\mathcal{N}$ can be used to construct a more representative training dataset for identifying the low-dimensional representation of the safe region. The SRL framework therefore focuses on ensuring safety of the real system when the initial policy is further improved during the learning process on the real system. Such a combination, on the one hand, increases the learning efficiency as the initial policy provided by transfer learning usually has a minimum expected performance and hence outperforms than a random policy. On the other hand, as the actual learning process is better imitated by the pre-training process, more useful training data points can be generated by using the proposed data generation method, which then leads to a better low-dimensional representation of the safe region for the SRL framework.

5.5.2. Limitations

One major limitation of the proposed data generation method is that, there is no clear guidance on how to select the parameter λ that balances the two sub-datasets. Currently, it is chosen by using experience and a rough estimation about the behaviour of the real system. Due to the uncertainty in such a selection, the performance of the SRL framework might not be the same as expected. Hence, a method that is able to find an optimal parameter λ according to the characteristics of the dynamical system and the learning task is desired. However, how to design such a method is a challenging problem.

Moreover, the performance of the proposed approach depends on the magnitude of discrepancy between the nominal and the real systems. Such a problem often happens when an approximated system model is used to predict the behaviour of the real system. Apparently, if the discrepancy is too large, then the derived low-dimensional representation of the safe region is less meaningful no matter how the training data are generated. Therefore, it is required that the simulation-to-reality gap should be bounded in a reasonable range, otherwise the introduction of the nominal system becomes less useful. Besides, if we are not confident about the reliability of the nominal system model, then a high value of parameter λ should be used in the data generation process, i.e., the training data are mostly drawn from the uniform distribution, such that high conservativeness is preserved in the SRL framework for keeping the system safe during the learning process.

5.6. Summary

In this chapter, we propose a data generation method that is able to provide representative training data for increasing the performance of the data-driven SRL framework. By considering the estimation of safety of real system states as a binary classification problem, we first analyze the influence of training data on the classification error. Then, we discuss the relationship between the classification error and the learning performance. Based on the analysis, a data generation method that divides the training dataset into two parts is presented accordingly. It utilizes a multivariate normal distribution and a uniform distribution to generate two sub-datasets, respectively. By adjusting the sizes of these two sub-datasets, a balance between finding a satisfying control policy and keeping the system safe is achieved. A three-link inverted pendulum example is provided to demonstrate the effectiveness of the proposed approach. The proposed data generation method not only improves the performance of the SRL framework, but also gives an insight about how different training data will affect the reliability of the safety estimates made via data-driven methods.

One possibility of the future work is to find a metric for quantifying the magnitude of discrepancy between the nominal and the real systems, such that a better estimate about the reliability of the nominal system model can be made. Then, the data generation process can be guided accordingly based on this estimate of reliability, such that the user is able to have more flexibility in choosing the tendency of the SRL framework.

6. Conclusion and Outlook

In this dissertation, we provide general SRL methods for complex dynamical systems by using MOR techniques. A conclusion regarding the proposed approaches is presented in this chapter, followed by an outlook about potential future research directions in this field.

6.1. Conclusion

While state-of-the-art SRL approaches given in recent studies demonstrate attractive results in low-dimensional dynamical systems, the computational limitations of these approaches hinder their applications to complex dynamical systems. To solve the challenging problem of employing SRL on dynamical systems with high-dimensional state space and highly nonlinear dynamics, we propose in Chapter 3 a general SRL framework that is based on physically inspired MOR. Provided with a predefined corrective controller, the SRL framework utilizes a supervisory control strategy that switches the actual applied action between the reinforcement learning-based controller and the corrective controller. The central idea is that, by finding a control invariant safe region, the learning-based controller has the flexibility in choosing its action if the system is within the safe region. Once the system approaches the boundary of the safe region, the corrective controller is activated to drive the system back to a safe state. However, computing an accurate safe region for complex dynamical systems is difficult. Therefore, by examining important physical features of the original system, we identify a simplified system model for predicting the safety of different original system states. Though a state mapping that maps each high-dimensional system state to a corresponding low-dimensional safety feature, a probabilistic estimate of safety is calculated accordingly, which is used by the supervisor to determine whether the current system state is safe or not. While the simplified system model enables the application of the SRL framework to complex dynamical systems, it is unavoidable that its prediction about the behaviour of the original system might be inaccurate. For having more reliable safety estimates, we hence propose an online adaptation method for modifying our belief about the safe region of the original system. By using a method called belief function theory, we generate two belief maps, one prior and one feedback belief maps, to consider the uncertainty contained in the simplified system model and the feedback data, i.e., the actual execution results of the corrective controller, respectively. Then, these two belief maps are fused via an operation referred to as weighted belief fusion to construct a more accurate estimate about the safe region. Such an estimate leads to a well-performed supervisor for the SRL framework, and realizes a satisfying balance between finding an optimal control policy and maintaining the safety of the dynamical system.

One major limitation of physically inspired MOR is that it requires a thorough under-

standing about the system dynamics so that representative physical features can be selected. However, in many practical control tasks, such an understanding is often not available or difficult to obtain. To overcome this problem, we propose a novel data-driven approach to identify a low-dimensional representation of the safe region for the SRL framework in Chapter 4. Inspired by transfer learning, we assume that the known information about the system dynamics provides us a nominal system model, which is used to generate training data for approximating the behaviour of the real system. By computing the probabilistic similarity between each pair of training data via a method called t-SNE, a realization of simplified states that best represents the training data is obtained. Such a realization, on the one hand, determines the state mapping, and on the other hand, leads to an initial estimate of the low-dimensional representation of the safe region. Due to the unknown and unmodelled part of system dynamics, there exists an inevitable mismatch between the nominal and the real systems. To account for this reality gap, a modified online adaptation method is also proposed. Each update iteration of the low-dimensional representation of the safe region consists of three steps. First, by learning a GPR model from the feedback data, the reliability of each training data is computed and an updated prior DSAF is derived. Second, a feedback DSAF is constructed from the feedback data for fully utilizing the valuable information about the real system behavior. Third, the two DSAFs are fused together for having a more reliable low-dimensional representation of the safe region. The SRL framework based on data-driven MOR is applicable to a wide range of dynamical systems and learning tasks, and provides an insight about how to safely extend reinforcement learning methods to real-world dynamical systems.

Like most of the data-driven approaches, the quality of training data affects the performance of the data-driven MOR significantly. For having a well-performed training dataset, we propose a data generation method in Chapter 5. By considering the estimation of safety as a binary classification problem, we analyze how training data will affect the classification error. Then, by simultaneously taking the classification error and the learning performance of the SRL framework into consideration, we design a data generation method that combines two distributions: one uniform distribution that is used to preserve conservativeness in the learning process, and one multivariate normal distribution for encouraging explorations for an optimal control policy. By balancing the sizes of the two sub-datasets, the proposed data generation method is able to produce training data that are most useful to the identification of the low-dimensional representation of the safe region, and hence leads to a better performance of the SRL framework. Moreover, it also provides an insight about how different training data will affect the accuracy in safety estimates that are obtained from data-driven approaches.

6.2. Outlook

Although the SRL methods proposed in this dissertation provide a possible solution to the challenging problem of safely extending state-of-the-art reinforcement learning methods to complex dynamical systems, there still exist multiple open problems for future researches. Some of these potential research directions are discussed as follows.

(i) **Combination of data-driven and projection-based MOR**
While data-driven MOR has the strength that it is applicable as long as a training

dataset is provided, it is usually not possible to result in a detailed mathematical relationship between the real and the simplified systems. Hence, it is difficult to quantify the reliability of the safety estimates obtained from the simplified system, e.g., what is the expected error when compared to the real system behaviour. In contrast, projection-based MOR is able to present a mathematical model that strongly connects the real and the simplified systems. However, projection-based MOR often requires solving partial differential equations for deriving a simplified system model, which is computationally infeasible for complex dynamical systems.

In recent studies, methods that utilize neural networks as an approximated solution to complicated equations, e.g., Hamilton–Jacobi–Isaacs equation [73], draw more and more attentions. By learning from observed data, a function approximator is constructed accordingly for estimating the possible solutions. This inspires us about the possibility of combining data-driven and projection-based MOR approaches. For example, by using data collected from the nominal system model, an approximator of the solutions to the relevant equations of projection-based MOR could be obtained. If a precise mathematical model that describes the connection between the real and the simplified systems can be derived, then a more reliable safety guarantee can also be proposed for the SRL methods. However, more investigations are needed for solving this challenging problem.

(ii) **Quantification of the discrepancy between nominal and real systems**

Simulation-to-reality gap is a frequently discussed problem when an approximated system model is used to predict the behaviour of the real system. Such a problem also occurs in the proposed SRL framework where a nominal system model is used to estimate the safety of real system states. Although many researches, both in machine learning and control theory, have investigated this problem, most of their solutions are task dependent and lack generality. Considering the usage of the nominal system in our SRL framework, one possible way to overcome the reality gap is to quantify the discrepancy between the nominal and the real systems. For example, this can be achieved by first measuring the trajectory similarities between the two systems. Then, by introducing a probabilistic model, e.g., a GPR model, among the similarities, the reliability of safety estimates for different system states can be predicted. The SRL framework therefore is able to give more flexibility to the learning-based controller when the current system state is believed to be safe with a high confidence, and restrict the exploration of the learning algorithm if it is not. However, how to design a metric for properly measuring the similarity and how to generalize the measurements with sampled system states to the entire system state space are both still open research problems.

(iii) **Extension to model predictive control**

In recent studies, instead of using a supervisory control strategy, robust model predictive control is also used for realizing SRL, e.g., in [66], [67]. As the real system model is included in computing the safety estimates, a bounded error is often provided by these approaches. However, if the dynamical system has a high-dimensional and highly

nonlinear dynamics, then it is computationally difficult to predict the system's behavior in a long horizon, which is unfortunately essential for model predictive control. A potential solution to this problem is, similar as in the proposed SRL framework, finding a simplified system model via MOR as an approximation for the real system. Then for saving computational costs, the predictions within the horizon can be made by using the two models: for timesteps that are close to the current step, the real system model can be used for having accurate estimates, while for far future, the simplified system model should be utilized for increasing the computational efficiency. However, further studies are required for properly integrating a simplified system model with model predictive control.

SOS Programming for Estimating the ROA

A.

We outline here a method described in [101], which is based on SOS programming to get an inner approximation of the ROA. Since this approximation is represented as a closed positive invariant subset of the ROA, it also provides an estimate of the safe region $\mathcal{S}$.

Let $\mathcal{P}$ be the set of all polynomials in n variables, then the set of SOS polynomials in n variables is defined as

$$\mathcal{P}_n = \{p \in \mathcal{P} \mid p \in \sum_{i=1}^{t} f_i^2,\ f_i \in \mathcal{P},\ i = 1, 2, \ldots, t\} \tag{A.1}$$

SOS programming solves an optimization problem to identify if a given polynomial is a SOS polynomial. It is shown that $p \in \mathcal{P}_n$ if and only if $\exists Q \succeq 0$ such that $p = z(x)^T Q z(x)$ with $z(x)$ as a vector of suitable monomials, and the existence of Q is a linear matrix inequality feasibility problem [125].

Consider a system described by (2.19) where $f(x)$ and $g(x)$ are represented as vectors of polynomials. For those $f(x)$ and $g(x)$ with non-polynomial forms, an approximated polynomial expression can be obtained through suitable Taylor expansion [81]. If the origin of the system is a locally asymptotically stable equilibrium point, the ROA can be estimated by the following calculations.

Suppose that a semialgebraic set $\mathcal{M}$ is given by a Lyapunov function $V(x)$ as $\mathcal{M} = \{x \in \mathcal{X} \mid V(x) \leq 1\}$, then an estimate of the ROA is found by maximizing a variable sized region $\mathcal{M}_\gamma = \{x \in \mathcal{X} \mid p(x) \leq \gamma\}$ subject to $\mathcal{M}_\gamma \subseteq \mathcal{M}$ [126], where $p(x)$ is a fixed and positive definite polynomial. This leads to the following optimization problem

$$\max_{V \in \mathcal{P}} \gamma, \tag{A.2}$$

with the requirement that following sets are all empty

$$\{x \in \mathcal{X} \mid V(x) \leq 0,\ l_1(x) \neq 0\}, \tag{A.3}$$

$$\{x \in \mathcal{X} \mid p(x) \leq \gamma,\ V(x) > 1\}, \tag{A.4}$$

$$\left\{ x \in \mathcal{X} \;\middle|\; \begin{array}{l} V(x) \leq 1,\ l_2(x) \neq 0, \\ \dfrac{\partial V}{\partial x}\left(f(x) + g(x)K(x)\right) \geq 0 \end{array} \right\}, \tag{A.5}$$

where l_1, l_2 are fixed positive definite and SOS polynomials which replace the constraints $x \neq 0$. In order to formulate SOS programming from (A.2-A.5), Positivstellensatz theorem [127] and $\mathcal{S}$-procedure [128] are employed, such that the previous optimization problem is transformed to the following form

$$\max_{V, s_1, s_2, s_3} \gamma, \tag{A.6}$$

with $V \in \mathcal{P}$, $s_1, s_2, s_3 \in \mathcal{P}_n$ such that

$$V - l_1 \in \mathcal{P}_n, \tag{A.7}$$

$$-((\gamma - p)s_1 + (V - 1)) \in \mathcal{P}_n, \tag{A.8}$$

$$-\left((1 - V)s_2 + \frac{\partial V}{\partial x}(f + gK)s_3 + l_2\right) \in \mathcal{P}_n. \tag{A.9}$$

Constraints as in (A.7-A.9) can be efficiently solved by semidefinite programming and an iterative algorithm can be used to obtain the approximation of the ROA. Note that, if the control policy is also given as a polynomial $K \in \mathcal{P}$, then the synthesis of the controller which enlarges the ROA can also be introduced into the optimization problem. For such cases, readers may refer to a two-step algorithm presented in [101] for more details.

SOS programming usually produces a conservative inner approximation of the ROA which can be used as an initial estimate of the safe region. However, for high-dimensional dynamical systems, performing such an estimation is generally infeasible, as the computational cost of SOS programming increases rapidly with respect to the dimensionality of system states.

B. Sensitivity Analysis of Parameter δ

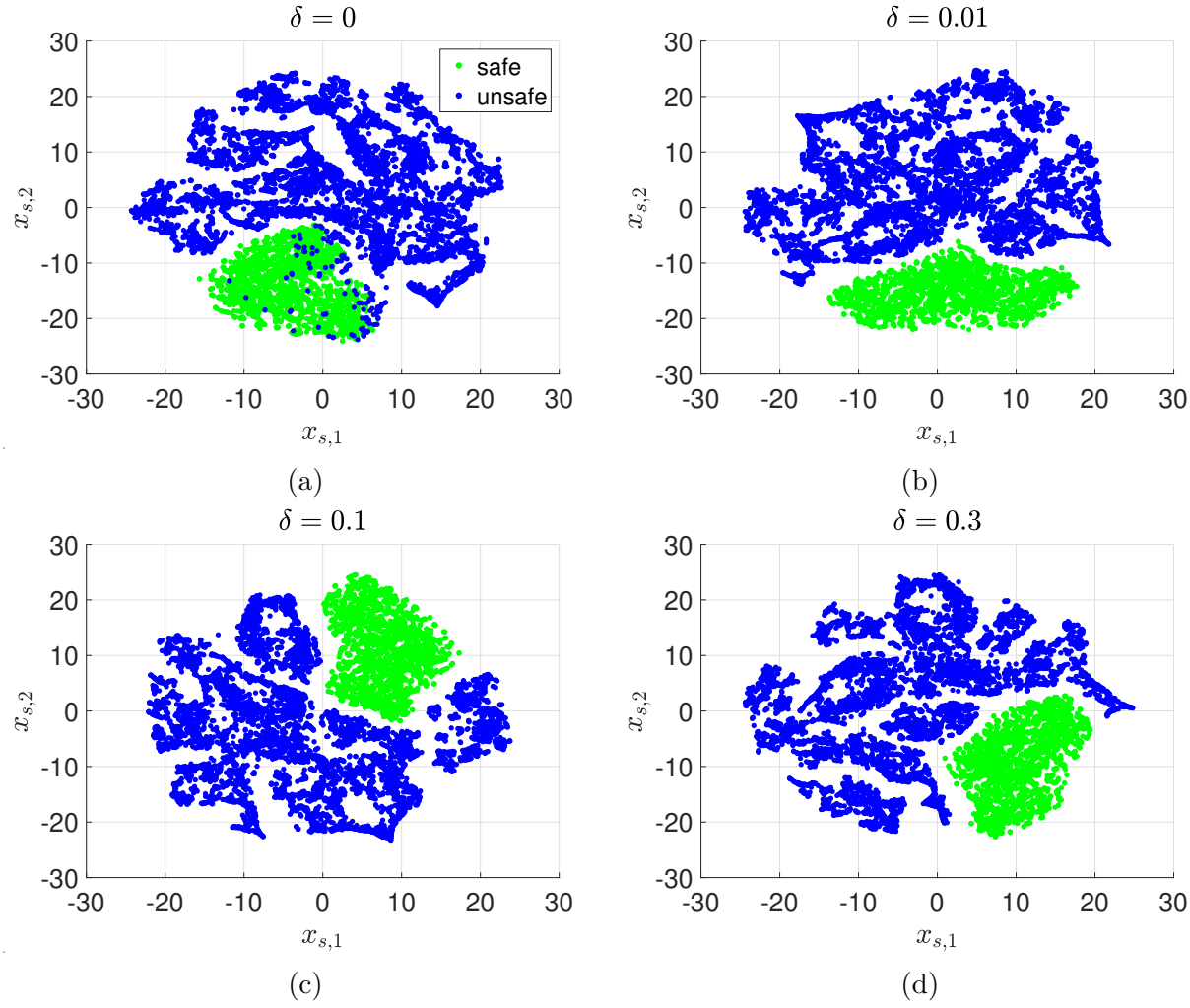


Figure B.1.: The initial realization of simplified states $x_s^1, \ldots, x_s^{k_t}$ obtained from t-SNE by using $\delta = 0$, $\delta = 0.01$, $\delta = 0.1$ and $\delta = 0.3$, respectively.

To analyze the influence of the parameter δ on the values of simplified states $x_s^1, \ldots, x_s^{k_t}$ obtained from t-SNE, we provide a sensitivity analysis in Fig. B.1, where the same training dataset $\mathcal{D}_{\text{train}}$ as the one given in the quadcopter example (see Section 4.4.1) is used.

It can be observed in Fig. B.1a that, when $\delta = 0$, i.e., the safety label is not considered in the computation of the distance Ω_{ij} (see (4.11)), there exist certain training data points that are not well separated according to their safety properties. This means that using only

the information contained in the pairwise trajectory distance ω_{ij} is not sufficient for finding a well-performed low-dimensional safety feature. To solve this problem, we add a constant parameter δ with a small value, e.g., $\delta = 0.01$, to (4.11). By doing this, the distance Ω_{ij} now also takes the difference in safety labels between training data points into consideration. Hence, as illustrated in Fig. B.1b, safe and unsafe training data points are better separated in the simplified state space.

However, further increasing the value of the parameter δ will not provide more benefits in finding a low-dimensional safety feature. As demonstrated in Fig. B.1c and Fig. B.1d with $\delta = 0.1$ and $\delta = 0.3$, the results are similar to those obtained with $\delta = 0.01$. In fact, a large value of the parameter δ leads to the similarity between safe and unsafe data points being measured almost entirely in terms of the difference in their safety labels. Therefore, the information contained in trajectories is highly likely to be ignored, which may reduce the representation power of the learned simplified states. To prevent this, it is suggested that the parameter δ is set to a small value, e.g., we use $\delta = 0.01$ in Chapter 4.

C. Computations of t-SNE

In the original version of t-SNE [119], Euclidean distances are used to compute a conditional probability $p_{j|i}$ as

$$p_{j|i} = \frac{\exp(-||x^i - x^j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x^i - x^k||^2/2\sigma_i^2)} \tag{C.1}$$

which stands for the probability that the high-dimensional state x^i picks the state x^j as its neighbor according to their probability density under a Gaussian distribution. However in our approach, to derive a realization of the low-dimensional safety feature that corresponds to the training dataset $\mathcal{D}_{\text{train}}$, we compute the conditional probability $p_{j|i}$ by using the proposed distance metric as given in (4.12). Any particular value of the parameter, i.e., the variance σ_i, results in a distribution P_i of the conditional probability $p_{j|i}$ over all other data points. A binary search is then performed to find the value of the variance σ_i, which produces a distribution P_i with a fixed perplexity that is selected by the user. The perplexity is defined as

$$\text{Perplexity}(P_i) = 2^{H(P_i)} \tag{C.2}$$

where $H(P_i)$ is the Shannon entropy of the distribution P_i measured as

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{C.3}$$

As stated in [119], the perplexity can be considered as a smooth measure of the effective number of neighbors.

To alleviate the identification problem caused by outliers, the similarity between two training data points is then defined by the joint probability p_{ij}

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2k_t} \tag{C.4}$$

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

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

$$q_{ij} = \frac{(1 + ||x_s^i - x_s^j||^2)^{-1}}{\sum_{k \neq i}(1 + ||x_s^i - x_s^k||^2)^{-1}} \tag{C.5}$$

where $||\cdot||$ is the Euclidean distance and we have $q_{ij} = 0$ if $i = j$. A heavy-tailed Student t-distribution is used here to measure the similarity. The values of simplified states $x_s^1, \ldots, x_s^{k_t}$

are determined by minimizing a cost function C given as

$$C = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{C.6}$$

where $\mathrm{KL}(\cdot)$ is the Kullback-Leibler divergence. P and Q are the joint probability distributions in the high-dimensional and low-dimensional state spaces, respectively. The cost function C represents how well the identified simplified states can reproduce the similarities between different training data.

The minimization is solved using a gradient descent method, where the gradient is computed as

$$\frac{\partial C}{\partial x_s^i} = 4 \sum_j (p_{ij} - q_{ij})(x_s^i - x_s^j)(1 + ||x_s^i - x_s^j||^2)^{-1} \tag{C.7}$$

The initial solution to the values of simplified states $x_s^1, \ldots, x_s^{k_t}$, denoted as $\mathcal{Q}^{(0)}$, is obtained by sampling points randomly from an isotropic Gaussian. To speed up the optimization and to avoid poor local minimum, the update of the solution is performed with a momentum term

$$\mathcal{Q}^{(t)} = \mathcal{Q}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Q}^{(t-1)}} + m^{(t)} \left(\mathcal{Q}^{(t-1)} - \mathcal{Q}^{(t-2)} \right) \tag{C.8}$$

where $\mathcal{Q}^{(t)}$ is the solution at iteration t, η is the learning rate, and $m^{(t)}$ is the momentum at iteration t. More implementation details of t-SNE are presented in [119].

Bibliography

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT press, 2018.

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, May 1996.

[3] N. J. Mackintosh, *The Psychology of Animal Learning*. New York, NY, USA: Academic Press, 1974.

[4] W. F. Prokasy and K. L. Kumpfer, "Electrodermal activity in psychological research," in W. F. Prokasy and D. C. Raskin, Eds. New York, NY, USA: Academic Press, 1973, ch. Classical Conditioning, pp. 157–202.

[5] A. Dickinson, "Animal learning and cognition," in N. J. Mackintosh, Ed. New York, NY, USA: Academic Press, 1994, ch. Instrumental Conditioning, pp. 45–79.

[6] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[7] I. H. Witten, "An adaptive optimal controller for discrete-time markov environments," *Information and Control*, vol. 34, no. 4, pp. 286–295, Aug. 1977.

[8] R. S. Sutton and A. G. Barto, "A temporal-difference model of classical conditioning," in *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, Jul. 1987, pp. 355–378.

[9] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[10] A. H. Klopf, *Brain function and Adaptive Systems: A Heterostatic Theory*. Cambridge, MA, USA: Air Force Cambridge Research Laboratories, 1972.

[11] ——, "A drive-reinforcement model of single neuron function: An alternative to the hebbian neuronal model," in *Proceedings of the AIP Conference on Neural Networks for Computing*, vol. 151, Mar. 1986, pp. 265–270.

[12] ——, "A neuronal model of classical conditioning.," *Psychobiology*, vol. 16, no. 2, pp. 85–125, Jun. 1988.

[13] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, 1989.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[15] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[17] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Feb. 2018, pp. 3207–3214.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013. [Online]. Available: `http://arxiv.org/abs/1312.5602`.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

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

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 770–778.

[22] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 290–298.

[23] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the Annual Conference of the International Speech Communication Association*, Sep. 2010, pp. 1045–1048.

[24] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *Proceedings of the IEEE International Conference on Computing, Communication and Automation*, May 2017, pp. 350–354.

[25] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, Jan. 2000.

[26] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, Nov. 2016.

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

[28] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, Apr. 2016.

[29] I. Akkaya, M. Andrychowicz, M. Chociej, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019. [Online]. Available: `http://arxiv.org/abs/1910.07113`.

[30] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, "Deep learning algorithm for autonomous driving using googlenet," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Jun. 2017, pp. 89–96.

[31] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, Apr. 2020.

[32] P. S. Thomas, "Safe reinforcement learning," Ph.D. dissertation, University of Massachusetts Amherst, 2015.

[33] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proceedings of the 2nd Annual Conference on Robot Learning*, Oct. 2018, pp. 561–591.

[34] H. Van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Apr. 2007, pp. 272–279.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015. [Online]. Available: `http://arxiv.org/abs/1509.02971`.

[36] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning*, Jun. 2016, pp. 1329–1338.

[37] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: Limitations imposed by unilateral contact," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Nov. 2016, pp. 8–15.

[38] S. Gugercin and A. C. Antoulas, "A survey of model reduction by balanced truncation and some new results," *International Journal of Control*, vol. 77, no. 8, pp. 748–766, Apr. 2004.

[39] A. Astolfi and R. Ortega, "Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems," *IEEE Transactions on Automatic control*, vol. 48, no. 4, pp. 590–606, Apr. 2003.

[40] G. J. Pappas and S. Sastry, "Towards continuous abstractions of dynamical and control systems," in *Proceedings of the International Hybrid Systems Workshop*, Oct. 1996, pp. 329–341.

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

[42] D. Laureiro-Martinez, S. Brusoni, N. Canessa, and M. Zollo, "Understanding the exploration-exploitation dilemma: An fmri study of attention control and decision-making performance," *Strategic Management Journal*, vol. 36, no. 3, pp. 319–338, Mar. 2015.

[43] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, Oct. 1957.

[44] S. P. Coraluppi and S. I. Marcus, "Risk-sensitive and minimax control of discrete-time, finite-state markov decision processes," *Automatica*, vol. 35, no. 2, pp. 301–309, Feb. 1999.

[45] R. A. Howard and J. E. Matheson, "Risk-sensitive markov decision processes," *Management Science*, vol. 18, no. 7, pp. 356–369, Mar. 1972.

[46] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 81–108, Jul. 2005.

[47] K. Driessens and S. Džeroski, "Integrating guidance into relational reinforcement learning," *Machine Learning*, vol. 57, no. 3, pp. 271–304, Dec. 2004.

[48] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, Jun. 2010.

[49] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," in *Proceedings of the 29th International Conference on Machine Learning*, Jun. 2012, pp. 1451–1458.

[50] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*. Dordrecht, Netherlands: Springer, Dordrecht, 2012.

[51] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, Aug. 2015.

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

[53] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proceedings of the 25th International Conference on Machine Learning*, Jul. 2008, pp. 144–151.

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

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

[56] S. Koos, J.-B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 122–145, Feb. 2012.

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

[58] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe multi-agent reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016. [Online]. Available: `http://arxiv.org/abs/1610.03295`.

[59] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning*, Aug. 2017, pp. 22–31.

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

[61] L. Prashanth and M. Ghavamzadeh, "Variance-constrained actor-critic algorithms for discounted and average reward mdps," *Machine Learning*, vol. 105, no. 3, pp. 367–417, Aug. 2016.

[62] V. S. Borkar, "Learning algorithms for risk-sensitive control," in *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*, Sep. 2010.

[63] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, Jan. 2017.

[64] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 803–832, Dec. 2002.

[65] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Proceedings of Advances in Neural Information Processing Systems*, Dec. 2018, pp. 8103–8112.

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

[67] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proceedings of Robotics: Science and Systems*, Jun. 2016, pp. 171–181.

[68] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[69] 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, May 2013.

[70] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Safe reinforcement learning using robust action governor," in *Proceedings of the 3rd Annual Learning for Dynamics and Control Conference*, Jun. 2021, pp. 1093–1104.

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

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

[73] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *Proceedings of the IEEE 56th Conference on Decision and Control*, Dec. 2017, pp. 2242–2253.

[74] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *Proceedings of the IEEE 55th Conference on Decision and Control*, Dec. 2016, pp. 4661–4666.

[75] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proceedings of Advances in Neural Information Processing Systems*, Dec. 2017, pp. 908–919.

[76] M. Köppen, "The curse of dimensionality," in *Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications*, Sep. 2000, pp. 4–8.

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

[78] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, Apr. 2020.

[79] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *SIAM Review*, vol. 60, no. 3, pp. 550–591, 2018.

[80] A. Forrester, A. Sóbester, and A. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, Dec. 2007.

[81] M. Posa, T. Koolen, and R. Tedrake, "Balancing and step recovery capturability via sums-of-squares optimization," in *Proceedings of Robotics: Science and Systems*, Jul. 2017, pp. 12–16.

[82] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM Review*, vol. 57, no. 4, pp. 483–531, 2015.

[83] M. Rathinam and L. Petzold, "A new look at proper orthogonal decomposition," *SIAM Journal on Numerical Analysis*, vol. 41, no. 5, pp. 1893–1925, 2003.

[84] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Philadelphia, PA, USA: SIAM, 2005.

[85] S. Klus, F. Nüske, P. Koltai, *et al.*, "Data-driven model reduction and transfer operator approximation," *Journal of Nonlinear Science*, vol. 28, no. 3, pp. 985–1010, Jan. 2018.

[86] A. Forrester and A. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, no. 1-3, pp. 50–79, Jan. 2009.

[87] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Optimal model management for multifidelity monte carlo estimation," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, A3163–A3194, Oct. 2016.

[88] G. Zhang, Y. Liu, and X. Jin, "A survey of autoencoder-based recommender systems," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 430–450, Apr. 2020.

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

[90] A. Marco, F. Berkenkamp, P. Hennig, *et al.*, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2017, pp. 1557–1563.

[91] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems," *IEEE Transactions on Neural Networks and Learning Systems*, early access, Sep. 2021.

[92] Z. Zhou, O. S. Oguz, Y. Ren, M. Leibold, and M. Buss, "Data generation method for learning a low-dimensional safe region in safe reinforcement learning," *arXiv preprint arXiv:2109.05077*, 2021. [Online]. Available: `http://arxiv.org/abs/2109.05077`.

[93] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Aug. 2013.

[94] B. Luo, Y. Yang, H.-N. Wu, and T. Huang, "Balancing value iteration and policy iteration for discrete-time control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 3948–3958, Mar. 2019.

[95] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement learning in continuous action spaces through sequential monte carlo methods," in *Proceedings of Advances in Neural Information Processing Systems*, Dec. 2007, pp. 833–840.

[96] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, May 2010.

[97] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992.

[98] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, Aug. 2009.

[99] C. Atkeson, "Nonparametric model-based reinforcement learning," in *Proceedings of Advances in Neural Information Processing Systems*, Dec. 1997, pp. 1–7.

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

[101] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *Proceedings of the IEEE 42nd Conference on Decision and Control*, Dec. 2003, pp. 4676–4681.

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

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

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

[105] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2018, pp. 2460–2465.

[106] H. Nickisch and C. E. Rasmussen, "Approximations for binary gaussian process classification," *Journal of Machine Learning Research*, vol. 9, no. Oct, pp. 2035–2078, Oct. 2008.

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

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

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

[110] ——, "Categories of belief fusion," *Journal of Advances in Information Fusion*, vol. 13, no. 2, pp. 235–254, Dec. 2018.

[111] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote Sensing of Environment*, vol. 62, no. 1, pp. 77–89, Oct. 1997.

[112] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. [Online]. Available: `http://arxiv.org/abs/1707.06347`.

[113] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors: A modular gazebo mav simulator framework," in *Robot Operating System*, Springer, 2016, pp. 595–625.

[114] 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," *arXiv preprint arXiv:1608.05742*, 2016. [Online]. Available: `http://arxiv.org/abs/1608.05742`.

[115] 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," *International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, Jul. 2012.

[116] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2018, pp. 6236–6243.

[117] S. R. Friedrich and M. Buss, "A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2017, pp. 3365–3372.

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

[119] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[120] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk, "Fréchet distance for curves, revisited," in *Proceedings of European Symposia on Algorithms*, Sep. 2006, pp. 52–63.

[121] T. Luukkonen, *Modelling and control of quadcopter*, Independent Research Project in Applied Mathematics, Aalto University, Espoo, Finland, Aug. 2011.

[122] R. Kumari and S. K. Srivastava, "Machine learning: A review on binary classification," *International Journal of Computer Applications*, vol. 160, no. 7, pp. 11–15, Feb. 2017.

[123] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1, pp. 151–175, Oct. 2010.

[124] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York, NY, USA: Springer, 2013.

[125] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. 293–320, Apr. 2003.

[126] Z. W. Jarvis-Wloszek, "Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-squares optimization," Ph.D. dissertation, University of California, Berkeley, 2003.

[127] G. Stengle, "A nullstellensatz and a positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, no. 2, pp. 87–97, Jun. 1974.

[128] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA, USA: SIAM, 1994.