



Title	Learning-Based Safe and Formal Control
Author(s)	橋本, 航
Citation	大阪大学, 2026, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/105452
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Doctoral Dissertation

Learning-Based Safe and Formal Control

Wataru Hashimoto

December 2025

Graduate School of Engineering,
The University of Osaka

Abstract

Achieving reliable, safe, and computationally efficient control of nonlinear and complex systems remains a central challenge in modern control engineering. Classical model-based approaches provide strong theoretical guarantees of stability and robustness, but depend on accurate models and well-characterized environments. In contrast, learning-based methods can flexibly adapt to data and changing conditions, yet reconciling their flexibility with rigorous guarantees and practical implementability remains challenging. This dissertation comprises three studies that address these challenges from complementary perspectives.

The first study develops a reference-free Learning Model Predictive Control (LMPC) framework that enhances control performance through iterative execution of MPC-based control. The proposed method employs Gaussian Process (GP) regression to model the uncertain components of the system dynamics and refines both the terminal set and terminal cost of the MPC optimization problem. These refinements are achieved using reachable sets computed from the reproducing kernel Hilbert space representation and the Lipschitz constants of the learned dynamics, accumulated over previous iterations. Theoretical analyses establish recursive feasibility of the MPC optimization problem and input-to-state stability (ISS) of the goal region, and a bound on the control performance of the closed-loop system at each iteration is also derived.

The second study addresses safe autonomous control in previously unseen environments by constructing Control Barrier Functions (CBFs) online from onboard sensor data. By combining Bayesian meta-learning with Bayesian surface reconstruction, the approach enables fast and data-efficient adaptation of the learned CBF to new environments and provides a probabilistic assessment of safety within a CBF-based quadratic program (CBF-QP) suitable for real-time implementation.

The third study focuses on control for temporal task specifications expressed in Signal Temporal Logic (STL). A learning framework is developed to train a single recurrent neural network (RNN) controller capable of handling multiple STL specifications. The proposed STL2vec embedding encodes STL formulas as vectors that capture their semantic relationships, allowing the controller to

generate task-conditioned policies that generalize across different objectives with improved computational and memory efficiency.

Contents

List of Figures	1
List of Tables	4
1 Introduction	5
1.1 Outline, Related Works, and Contribution	8
1.1.1 Chapter 2	8
1.1.2 Chapter 3	10
1.1.3 Chapter 4	13
2 Robust Learning-Based Iterative Model Predictive Control for Unknown Non-Linear Systems	16
2.1 Preliminaries	17
2.1.1 System description	17
2.1.2 Gaussian process regression	17
2.1.3 Propagation of the confidence regions	18
2.2 Problem Statement	21
2.3 Proposed Method	24
2.3.1 Collection of terminal sets $\mathcal{SS}^j$	26
2.3.2 Backward enlargement of reachable sets	27
2.3.3 Terminal cost Q^j	29
2.4 Properties	29
2.4.1 Recursive feasibility	29
2.4.2 Input to state stability of the invariant set $\mathcal{O}$	31
2.4.3 Performance bound	33
2.5 Practical Considerations	34
2.5.1 Control scheme with GP update	34

2.5.2	Construction of the initial sets	35
2.6	Case Study	36
2.6.1	Double integrator	37
2.6.2	Vehicle navigation	38
2.7	Summary	38
3	Data-Efficient Safe Learning and Control with On-Board Sensors: Bayesian Meta-Learning and Barrier Function Based Approach	40
3.1	Problem Statement	41
3.2	CLF-CBF-QP	42
3.3	Bayesian Meta-Learning of CBF Candidate	43
3.3.1	Dataset construction	44
3.3.2	Bayesian meta-learning of the function h_{IS}	45
3.4	Online Control Execution	49
3.4.1	The update scheme of each buffer $\mathcal{B}_i$	50
3.4.2	Theoretical results	50
3.5	Case Study	51
3.5.1	Meta learning of unsafe regions	51
3.5.2	Control execution	52
3.6	Summary	54
4	STL2vec: Signal Temporal Logic Embeddings for Control Synthesis With Recurrent Neural Networks	57
4.1	Preliminaries	57
4.1.1	System description and notations	57
4.1.2	Signal Temporal Logic	58
4.2	Problem Statement	59
4.3	Summary of Word2vec (skip-gram)	62
4.4	Proposed Scheme	63
4.4.1	Training STL2vec	64
4.4.2	Training RNN	67
4.4.3	Some comparisons between the proposed approach and the approach of learning the RNNs one by one	68
4.5	Case Study	69

4.5.1	Training STL2vec	69
4.5.2	Training RNN	71
4.6	Summary	75
5	Conclusion	77
5.1	Summary	77
5.2	Future Work and Outlook	78
	Acknowledgments	80
	Publication	81
	References	83

List of Figures

- 2.1 The monotonicity property of the mapping $\tilde{m}$. If $\Phi_{t-1}^1 \subseteq \Phi_{t-1}^2$ holds for two ellipsoids $\Phi_{t-1}^1 = \mathcal{E}(p_{t-1}, P_{t-1}^1)$ and $\Phi_{t-1}^2 = \mathcal{E}(p_{t-1}, P_{t-1}^2)$, the ellipsoids mapped by $\tilde{m}$ with any control input $u_{t-1} \in \mathcal{U}$ hold the relationships $\Phi_t^1 \subseteq \Phi_t^2$ and $p_t^1 = p_t^2$ 21
- 2.2 An example of the execution in iteration 1 when the initial control sequence and the corresponding sequence of confidence regions (blue ellipsoids) are given by $\hat{u}_{0:6|0}^{0,*}$ and $\Phi_{1:6|0}^{0,*}$ as shown in the left above figure. The horizon length is set to $N = 2$ in this example. At time 0 in iteration 1 (middle above figure), the LMPC optimization (2.21) at $x_0^j = x_s$ is solved, and the optimal sequence of inputs $\hat{u}_{0|0}^{1,*}, \hat{u}_{1|0}^{1,*}$ and the corresponding confidence regions $\Phi_{1|0}^{1,*}, \Phi_{2|0}^{1,*}$ are obtained to satisfy the terminal constraint (in this case, $\Phi_{2|0}^{1,*} \subseteq \Phi_{2|0}^{0,*}$). Then, the first control input $u_0^1 = \hat{u}_{0|0}^{1,*}$ is applied to the system and the next state x_1^1 is observed. This process is repeated until the prescribed time T^1 is reached. After iteration 1, the terminal set $\mathcal{SS}^0$ and cost Q^0 are updated by following the procedures in subsections 2.3.1, 2.3.2 and 2.3.3. The procedures in the later iterations similarly follow. 24
- 2.3 Backward Enlargement of the reachable sets. In this example, we first assume that the optimal control inputs $\hat{u}_{t:t+2|t}^{j,*}$ and reachable sets $\hat{\Phi}_{t:t+3|t}^{j,*}$ with $\hat{\Phi}_{t+3|t}^{j,*} \subseteq \Phi_{t^*+k^*|t^*}^{i^*}$ are obtained by solving the problem (2.21) as shown in the left above figure. Then, the enlargement of the sets $\hat{\Phi}_{t+2|t}^{j,*}$ and $\hat{\Phi}_{t+1|t}^{j,*}$ are performed in this order by solving (2.26) with $\Phi_{t+3|t}^{j,*} = \Phi_{t^*+k^*|t^*}^{i^*}$. With this scheme, we can enlarge the sets without breaking the inclusion relationships $\tilde{m}(\Phi_{t+k|t}^{j,*}, \hat{u}_{t+k|t}^{j,*}) \subseteq \Phi_{t+k+1|t}^{j,*}, \forall k \in \{1, 2\}$ by effectively using the fact that the terminal ellipsoid $\hat{\Phi}_{t+3|t}^{j,*}$ satisfies the terminal constraint (2.21g) with some margin. 28

2.4	The explanation of the recursive feasibility property when $N_t^j = N_t^{j+1} = N$. The blue, green, and red ellipsoids represent the set in $\mathcal{SS}^{j-1}$ ($\Phi_{t^*+k^* t^*}^{i^*}$ is the terminal set selected at time t in iteration j and $\Phi_{t^*+k^*+1 t^*}^{i^*}$ is the corresponding one step forwarded set), the optimal sequence of ellipsoids calculated at time t in iteration j , and the sequence of ellipsoids generated by the control inputs $\hat{u}_{t+1 t}^{j,*}, \dots, \hat{u}_{t+N_t^j-1 t}^{j,*}, u_{t^*+k^* t^*}^{i^*}$ at time $t+1$ from the initial ellipsoid $\tilde{\Phi}_{t+1 t+1}^j = \mathcal{B}(\hat{p}_{t t-1}^{j,*}, x_t^j - \hat{p}_{t t-1}^{j,*})$, respectively. From the monotonicity property of the propagation of the ellipsoids summarized in Theorem 2 and the definition of the terminal set discussed in subsections 2.3.1 and 2.3.2, the above sequence of control inputs is a feasible solution of the problem (2.21) at time $t+1$ (red ellipsoids are within a blue or green ellipsoid and centers of these ellipsoids are matched).	30
2.5	The actual trajectories obtained by executing the proposed method. We can see that the performance of the closed-loop system is improved along with the iterations. . . .	36
3.1	The construction of data points from noisy sensor measurements: The gray dots are the points on the surface detected by the sensors. The green, blue, and red dots represent the points that have distance $2\Delta, \Delta, -\Delta$ from the surface (negative value means inside of the obstacle) which are calculated by using surface normal approximation explained in subsection 3.3.1 (we show the case of $n_- = 1$ and $n_+ = 2$).	44
3.2	The summary of the proposed approach: The proposed method consists of <i>offline meta-training</i> and <i>online control execution</i> . The offline training follows the procedures in subsection 3.3.2. In each iteration, we first sample J obstacles from the distribution $p(\xi)$ and construct the datasets $D_{\xi_j}^{\text{tr}}$ and $D_{\xi_j}^{\text{ts}}$ for each of these obstacles. Then, the loss (3.10) of the prediction made by the model adopted from $D_{\xi_j}^{\text{tr}}$ is evaluated for test data $D_{\xi_j}^{\text{ts}}$ and the parameters are updated with the gradients of the loss. In the online control execution, the control input is generated by CLF-CBF-QP (3.5) with the CBF candidates obtained through the adaptation of the posterior parameters (3.8) and the probabilistic bounds (3.14). The dataset for the adaptation is constructed using the sensor measurements obtained during the operation by following procedures in subsection 3.3.1 and the data selection scheme discussed in subsection 3.4.1.	45
3.3	PyBullet objects in case (II). The figures in the first and second lows are the GUI of the objects and corresponding cross sections at x - y coordinates, respectively. The surfaces with the smallest and largest sizes are shown by the black and blue lines, respectively.	52

3.4	A few examples of the prediction on surfaces: The black and blue lines show the actual surface and the mean prediction of the surface, respectively. The blue shaded areas show the regions between the 0-level sets of the upper and lower 90% bounds of h_{IS} obtained from the discussion in subsection 3.3.2, and the black cross marks show the data points on the surface used for the adaptation.	53
3.5	Left and Middle: NLLs for predictions of the proposed meta-learning scheme and GP case against the number of data points for the cases (I) and (II) (the number of data points in the figure means the number of surface points and the actual number of data points is $n_+ + n_- + 1$ ($= 7$) times of that). Right: the time required for the training against the number of data points. The means and $3 - \sigma$ intervals are plotted in the figures.	54
3.6	Environments for testing the control performance.	55
3.7	The control execution in Env 1 with the proposed method (above) and GP case (below). The red crosses, heat maps, blue solid lines, black solid lines, and black dotted lines represent data points on the surface, the values of CBF, 0-level sets of 95% lower bounds of CBF, actual surfaces, and robot trajectories, respectively. The CBF is updated every 5 [s].	56
4.1	The structure of RNN.	60
4.2	Skip-gram model.	62
4.3	Overview of the proposed approach.	64
4.4	Regions and sub-regions in a 2D space.	70
4.5	The average of robustness values for each control scheme (proposed method and schemes (A1)-(A3)). The average has been taken by testing the controllers for every 10 epochs with 30 initial states randomly sampled from the initial region $\mathcal{X}_0$	73
4.6	The testing result for 10 initial states and the 3 specifications: (i) $F_{[0,15]} G_{[0,5]} \text{Reg}(1, 1) \vee F_{[0,15]} G_{[0,5]} \text{Reg}(3, 2)$, (ii) $F_{[0,10]} \text{Reg}(4, 1) \wedge F_{[11,20]} \text{Reg}(3, 2)$, (iii) $F_{[0,20]} \text{Reg}(4, 4) \wedge (\neg \text{Reg}(4, 4) U_{[0,20]} \text{Reg}(2, 3))$. Results are shown for both (a) the proposed scheme and (b) approach (A3).	74

List of Tables

2.1	Performance cost of the proposed control scheme at each iteration (double integrator)	37
2.2	Performance cost of the proposed control scheme at each iteration (vehicle navigation)	38
3.1	The CSEs between the robot position and the goal position	52
4.1	STL candidate specifications	70
4.2	The time required to obtain STL2vec (in sec)	71
4.3	Cosine similarities of vector representations of the STL candidate specifications . . .	71
4.4	Some alternative approaches	72
4.5	Run time needed to reach several average of robustness (in sec)	72
4.6	Total number of parameters in each method	75

Chapter 1

Introduction

Designing effective policies for control and decision-making—determining appropriate actions based on system states to achieve desired objectives—remains a central challenge in engineering and intelligent autonomous systems [1, 2]. Classical model-based control theory offers a rigorous foundation for analyzing and guaranteeing stability, robustness, and performance of a closed-loop system [3–5]. However, its success fundamentally relies on the availability of accurate system models and consistent knowledge of the operating environment. When models are misspecified or the environment undergoes unanticipated changes, the nominal guarantees of stability and performance can deteriorate rapidly [6]. To extend the reach of classical control to increasingly complex and uncertain domains such as autonomous driving, robotic manipulation, and large-scale energy systems, it is essential to develop design principles that ensure reliable and adaptive operation even when precise models or fixed operating conditions cannot be assumed.

In recent years, rapid progress in the machine learning has positioned data-driven approaches as powerful tools for addressing the modeling and design limitations of conventional control methods. Within the learning community, Reinforcement Learning (RL) has emerged as a general framework for sequential decision-making, where an agent learns a control policy that maximizes cumulative rewards through trial-and-error interaction with an initially unknown environment [7]. This formulation enables learning directly from experience without relying on explicit analytical models, and has demonstrated remarkable flexibility in high-dimensional, nonlinear, and partially observed settings [8]. Notably, RL has achieved superhuman performance in challenging domains such as Go and real-time strategy games [9, 10]. However, these achievements have yet to fully translate to safety-critical physical systems such as robotics, where issues of high sample complexity and the absence of formal guarantees for safety and stability remain major barriers to reliable deployment [11, 12]. Addressing these safety concerns is a central goal of the growing subfield of safe reinforcement learning (Safe

RL) [12], yet bridging the gap between empirical performance and the formal verification required by safety-critical systems remains a key obstacle.

In parallel, the control community has sought to augment classical model-based control theory by integrating adaptation and learning techniques. A key line of research in this direction is data-driven system identification, which aims to learn dynamical models directly from experimental or operational data, thereby enabling controller synthesis even when accurate physical models are unavailable [13]. Recent advances in neural-network-based identification and differentiable dynamics learning have further enhanced the expressiveness and adaptability of learned models for complex nonlinear systems [14, 15]. Building on these developments in model learning, a significant research effort has focused on addressing the inherent uncertainty associated with data-driven models during both design and operation. Instead of relying solely on nominal predictions, these approaches explicitly characterize uncertainty, often through probabilistic learning frameworks, and incorporate this information into reasoning about system behavior. For example, probabilistic models such as Gaussian processes have been used to identify reliable operating regions and to expand these regions cautiously as additional data reduce uncertainty, ensuring that the system operates within confidence-certified bounds throughout the learning process [16–18].

In addition to learning dynamical models and their uncertainty, recent work has investigated how data can refine key structural components of the controller itself, including performance objectives, feasible-set descriptions, and feedback parameterizations. These approaches preserve a model-based architecture while treating central design elements as tunable parameters that can be optimized to better reflect real system behavior. Early examples include self-tuning and adaptive optimal control schemes that update feedback gains or estimator parameters online in response to observed performance [19, 20]. More recent contributions employ Bayesian optimization and other data-efficient black-box optimization methods to automatically tune quadratic performance weights, cost-shaping parameters, and feedback gains based on experimental data [21–24].

Another prominent research direction in learning-based control literature is learning control certificate functions, such as the Control Lyapunov Function (CLF) [25], the Control Barrier Function (CBF) [26], and contraction metric [27]. A certificate function provides a formal means to verify that a control system satisfies desired properties—most commonly stability, safety, or performance. A CLF ensures stability by guaranteeing the existence of a control input that decreases a Lyapunov-like energy function, driving the system toward a desired equilibrium. Conversely, a CBF ensures safety by maintaining forward invariance of a predefined safe set through inequality constraints that prevent the system from leaving that region [28–31]. A contraction metric, in turn, certifies incremental stability

or exponential convergence rates by ensuring that trajectories converge to each other [27]. The primary challenge lies in constructing valid certificate functions for nonlinear or high-dimensional systems, as doing so often requires strong analytical insight into system dynamics. This task becomes particularly difficult when the environment or safe set is not known a priori. To overcome this limitation, recent research has focused on learning certificate functions directly from data [32–35]. These methods typically employ neural networks or optimization-based representations to enforce Lyapunov or barrier inequalities at sampled states, sometimes with robustness margins or counterexample-guided refinement, i.e., iteratively searching for states that violate the certificate conditions and adding them to the training/constraint set to refine the candidate certificate.

Beyond certificate-based approaches, formal methods grounded in temporal logic—including Linear Temporal Logic (LTL) [36] and Signal Temporal Logic (STL) [37]—provide a complementary framework for specifying and verifying complex, time-dependent behaviors of dynamical systems [38]. Whereas Lyapunov- and barrier-based certificates impose continuous-time conditions that ensure stability and safety, temporal logic captures higher-level task requirements such as ordered events, timing constraints, and persistent objectives. LTL encodes qualitative logical relations among propositions over time, while STL extends this capability to real-valued, continuous-time signals and supports quantitative reasoning through robustness scores [37, 39]. In control synthesis, such robustness metrics can be incorporated into optimization or MPC formulations to enforce temporal and safety constraints in a unified manner [40–42]. Motivated by these developments, learning-based approaches have recently been explored for synthesizing control policies that satisfy temporal logic specifications directly from data [43–45]. In these methods, neural networks are trained to maximize differentiable approximations of STL robustness or minimize the deviation from the expert demonstrations, enabling the policy to acquire temporal reasoning abilities while accounting for nonlinear dynamics. Once trained offline, the resulting controller can generate appropriate inputs in real time without repeatedly solving nonconvex optimization problems in online control execution.

This dissertation builds on the research themes introduced above and develops new methodologies that address their limitations in the context of learning-based control. The detailed outlines, related works, and specific contributions of each chapter are presented in the following section.

1.1 Outline, Related Works, and Contribution

1.1.1 Chapter 2

Model Predictive Control (MPC) [46] is one of the most commonly used control methods that has a wide range of industrial applications. The ability to systematically cope with multivariate plants and system constraints has made MPC an attractive control strategy. However, since the control inputs are determined by solving a finite time horizon optimal control problem in a receding horizon manner, the resulting state trajectory is generally a sub-optimal solution to the original infinite time optimal control problem. To address this optimality issue, arising from the short-sightedness of MPC, some previous works consider improving the control performance by iteratively executing control with finite-horizon MPC. Exploiting such a framework is especially beneficial for applications that are required to iteratively execute the same task, e.g., robots for assembling. Under the assumption that the reference signal is given in advance, many works, such as [47, 48] achieve zero tracking error with the iterative MPC schemes. The authors of the works [49, 50] proposed a different iterative MPC approach, called Learning Model Predictive Control (LMPC) that considers improving the terminal components (i.e., terminal cost and terminal set) of the MPC problem, by using the experiences in the previous iterations. In these works, the reference signal is not required, which improves the applicability of the method.

In addition to the optimality issue, the lack of an accurate prediction model is also a major practical challenge for MPC implementations [6]. Thus, the ways to learn the system dynamics with data, and incorporate it into the MPC optimization problem are studied in the vast literature [6]. Specifically, Gaussian Process (GP) [51, 52] is one of the most popular learning methods in this context [18, 53–57]. Since the GP can capture the uncertainty in the prediction and is equipped with formal probabilistic bounds on the prediction [58, 59], it can be used to consider or ensure robust constraint satisfaction of the resulting controller [55].

In this study, we investigate a reference-free iterative MPC scheme for unknown nonlinear systems. Our method extends the LMPC framework [49], specifically adapted to scenarios where the system dynamics are not known apriori and learned from data. This extension is not trivial since the terminal constraint in [49] requires the terminal state to coincide with one of the states that are visited in past iterations, which is impossible in our setting due to the state-dependent uncertainty in the prediction model. Since the dynamics model always includes some prediction error and the dynamics may change during the control execution, it is crucial to develop such a method to enhance the LMPC framework’s applicability. In the proposed method, we first consider learning the controlled dynamics

by GP regression and derive forward-reachable sets of system states with respect to a sequence of control inputs using the Reproducing Kernel Hilbert Space (RKHS) augment. Then, the terminal set and cost in the MPC problem are defined based on the reachable sets that are calculated and collected in the previous iterations. To iteratively improve the terminal components, we leverage and extend the ideas in the works [49, 50]. Specifically, how to construct the terminal components for the unknown dynamics setting to get a desirable performance is considered. The proposed MPC scheme meets several desirable properties such as recursive feasibility and Input-to-State Stability (ISS) of the goal region. We also derive the performance bound of the resulting closed-loop system in each iteration under certain assumptions. The case study shows that the proposed iterative MPC scheme can effectively improve the performance cost along with the iterations.

Related works and contributions: Learning-based approaches for MPC have been actively studied over the last few decades, and many attempts have been made in the literature [6, 60]. Most research efforts in learning-based MPC have focused on the automatic adaptation of the system model. Specifically, how to guarantee robust constraint satisfaction with learned models has gained significant attention in the literature. The robust adaptive formulations that use set membership estimation to tighten the constraints are introduced in the works such as [61, 62]. Machine learning-based approaches are also extensively studied in a variety of works [18, 53–57, 63–65]. Specifically, the works [18, 53–57] use the GP, which can express uncertainty in the prediction and enable controllers to evaluate the probability of violating the constraints. While the works [18, 53, 54] rely on the approximated prediction of uncertainty propagation of the GP such as statistical moment matching, the authors of the work [55] derive ellipsoidal sets that are ensured to include the actual system states with high probability based on Reproducing Kernel Hilbert Space (RKHS) augment and use them to tighten the constraints in MPC problem. The works [56, 57] guarantee closed-loop stability of the GP-based robust MPC.

The potential for learning and data-driven adaptation in MPC extends further than just enhancing model accuracy. Specifically, how to improve the closed-loop performance of MPC controllers by adjusting parameters within MPC formulation with collected data is considered in the literature. Bayesian optimization for optimal control [66], inverse optimal control [67], and a combination of reinforcement learning and MPC [68] are examples of such a method. This study is mainly built on the seminal works by Ugo Rosolia and Francesco Borrelli [49, 50], which propose a way to effectively incorporate the obtained data in the previous control executions to refine the terminal components of the MPC problem. With this scheme, the infinite-time optimal control problem with state and input constraints is guaranteed to be solved by iteratively performing finite-time horizon MPC for

known linear systems [50] and the monotonically non-increasing property of the performance cost is guaranteed for known nonlinear systems [49], under the assumption that one or more feasible (not optimal) solution to the control problem is given in advance. Moreover, the desired properties for MPC, such as recursive feasibility and stability of the closed-loop system, are also guaranteed with the learned terminal components. This method is extended to the case of uncertain linear systems [69, 70], nonlinear probabilistic systems [71, 72], multi-agent systems [73], certificate function based approach [74], and multi-modal setting [75], and tested on a number of challenging applications such as autonomous vehicles [76] and surgical robots [71]. However, all of these studies assume that the system dynamics are known or uncertain, but the noise distribution is given, which motivates our work.

The contributions of our work in the literature are the following three hold: First, we consider how to iteratively improve the terminal components of the MPC problem with the learned dynamics and show the resulting MPC formulation meets desirable properties such as recursive feasibility and Input-to-State Stability (ISS) of the target region under certain assumptions. By further adding conditions to the proposed control scheme, we can also show the system states converge to the target region. Moreover, we analyze the performance bound along with the control execution in each iteration. The conditions required for the learned dynamics to be incorporated into the proposed scheme are also clarified. To the best of the author’s knowledge, this is the first work that considers an LMPC method (method aiming to iteratively improve the terminal components) for unknown nonlinear systems, while adaptive control strategies for linear systems are explored in [77]. Second, we introduce the technique called Backward Enlargement, which enlarges the confidence regions obtained in the previous iterations backward in time to improve the method’s applicability (for details, see subsection 2.3.2). Third, compared to the MPC strategies with the uncertainty propagation techniques such as [55, 63] without iterative control schemes, our work can effectively use the reachable sets observed in the past iterations and ensure the feasibility of control problems that require larger execution time steps than the MPC horizon, discuss the stability property of the target region and the performance cost bound. Lastly, we test the proposed method with the simulation study on two control problems and see that the proposed method can improve the performance cost along with the iterations.

1.1.2 Chapter 3

Ensuring safety while achieving certain tasks is a fundamental yet challenging problem that has drawn significant attention from researchers in the control and robotics communities over the past few decades. In the control community, such safety requirements are often handled by imposing

corresponding state constraints on an optimal control problem. Model Predictive Control (MPC) [46] and control synthesis methods based on the certificate functions, such as Control Lyapunov Function (CLF) and Control Barrier Function (CBF) [25, 26, 30] are notable examples of such a method. Desirable theoretical results for these methods, such as feasibility, stability, and safety are offered for known system dynamics [25, 26, 30, 46] or noisy dynamics with known noise distribution [78–80].

The CBF is a prominent tool to impose forward invariance of a safe set in state space. If a dynamical system is control-affine, Quadratic Programming formulation called CBF-QP [30] is available, which can produce safe control inputs much faster than many other control methods, including MPC, and enables real-time implementations for challenging applications such as autonomous vehicles [81] and bipedal locomotion [82]. However, the majority of the existing works presuppose the availability of a valid CBF that represents safe and unsafe regions in state space. This assumption, however, cannot be maintained if a robot is expected to operate autonomously in unknown or uncertain environments. In such situations, it is essential for a robotic system to automatically identify unsafe regions on the fly, employing data from sensory devices. Thus, in this study, we consider a learning-based method for constructing a CBF based on online measurements obtained from onboard sensors such as LiDAR scanners.

In the previous studies regarding this line of the topic (e.g., [83–85]), the robotic system iteratively improves the prediction of the CBF by collecting online sensor measurements without any prior knowledge about the environment or learning task. However, in such cases, even a slight change in the environment necessitates a complete retraining process, which is inefficient and requires a substantial amount of data. To efficiently and effectively learn CBF, adapting it to a new environment with minimal or small amount of training data, we suggest implementing meta-learning [86, 87], a technique that allows an agent to *learn how to learn*, as opposed to creating it from scratch. During the meta-training process, the model parameters are trained with data collected from various different environments, aiming to learn common underlying knowledge or structure of the learning task. The parameters obtained through meta-learning are then utilized as initial parameters for the online phase in a new environment and updated with online measurements in the environment. In this study, we specifically employ a learning scheme combining the recently developed Bayesian meta-learning method [88] which is equipped with probabilistic bounds on the prediction [89], and the Bayesian surface reconstruction method [90–92] that can learn the unsafe regions from noisy sensor measurements. With this method, we can learn CBF and synthesize a safe controller with few amount of data and provide a probabilistic guarantee on the resulting controller.

Related works on learning and CBF : Learning-based approaches for certificate functions such

as CBF and CLF are one of the active research topics in the area of intersections between control and learning [93]. In the following, we summarize the recent progress in the topic of learning and CBF. The authors of the works [94–99] used CBF for active safe learning of uncertain dynamics. The works [94, 95] consider machine learning approaches to safely reduce the model uncertainty and show the methods yield empirically good performances, while the Gaussian Process (GP) is also used in [96–99] and rigorous theoretical results for safety requirements are offered under certain assumptions. CBF is also used in Imitation Learning (IL) and Reinforcement Learning (RL) [100–104] to account for safety concerns.

While the above studies assume a valid CBF is given and consider leveraging it to impose safety conditions, several previous studies also consider parameterizing CBF itself by Neural Network (NN) and learning it to be a valid barrier function [83–85, 105–108]. The work [105] considers jointly learning NNs representing CBF and CLF as well as a control policy based on safe and unsafe samples. The method proposed in [106] considers recovering CBF from expert demonstrations and theoretically guarantees that the learned NN meets the CBF conditions under the Lipsitz continuity assumptions. In [83], GP is also used to synthesize CBF. The works most related to this study are [84, 85], which propose ways to learn CBF with sensor measurements. In [84], Support Vector Machine (SVM) classifier representing CBF is trained with safe and unsafe samples constructed by using the LiDAR measurements. Similarly, the authors of the work [85] present a synthesis method using Signed Distance Function (SDF).

Related works on meta-learning and control : Meta-learning, also known as “learning to learn,” is a technique capable of learning from previous learning experiences or tasks in order to improve the efficiency of future learning [86]. Meta-learning has recently become increasingly popular in the control literature. In several previous studies, such as [89, 109], meta-learning is used to learn system dynamics to enable quick adaptation to the changes in surrounding situations. In these works, the trained models are incorporated into MPC, which requires relatively heavy online computation. Meta-learning is also used in policy learning in the context of adaptive control [110] and reinforcement learning [86, 111].

Contributions: Our contributions compared to the existing methods are threefold: First, compared to the methods that synthesize CBF with sensor measurements [83–85] which rely on traditional supervised learning, the proposed meta-learning scheme can effectively use past data collected from different environments and produce a prediction of CBF with few amount of data. Consequently, the resulting control scheme realizes less conservative control performance with few amount of data (small number of online CBF updates) as we see in the case study. Second, different from the previous

NN-based CBF synthesis methods [84, 85, 105, 107, 108], our method can readily take into account uncertainty in data and calculate formal probabilistic bounds on CBF. Based on this, we provide a probabilistic safety guarantee on the proposed control scheme. Although the work [85] takes into account the prediction error in Signed Distance Function (SDF), such an error bound is assumed to be given and actual computation is not provided. Third, the learned CBF can readily be incorporated into ordinary QP formulation, which can be efficiently solved. Thus, our method can produce control input much faster than the previous methods regarding meta-learning, and MPC, such as [89, 109].

1.1.3 Chapter 4

Recently, formal methods have attracted much attention in control to address complex and temporal objectives such as periodic, sequential, or reactive tasks, going beyond the traditional control objectives like stability and tracking. Temporal logics such as linear temporal logic (LTL)[36], metric temporal logic (MTL)[112], and signal temporal logic (STL)[37] allow us to write down formal descriptions for these specifications.

In this study, we focus on dealing with the STL specifications, which can specify the temporal properties of real-valued signals. One of the notable advantages of using the STL specifications in control is that we have access to the quantitative semantics, called *robustness*, which measures how much a signal satisfies the STL specification. The robustness has been used for control purposes in many existing works [40, 113–118]. The authors of [40, 113, 114] proposed a scheme of encoding STL constraints or robustness of the STL specifications by mixed-integer linear constraints, and these are utilized to employ Model Predictive Control (MPC). To alleviate the computational burden of mixed-integer programming, the works [115–118] proposed the concept of *smooth* robustness, allowing us to solve the maximization problem of the robustness based on gradient-based algorithms.

Aiming to further improve the scalability and meet real-time requirements, neural network (NN) based feedback controller was employed for STL tasks [43–45, 106, 119]. Although the learning procedure itself may be time-consuming, once this computation can be done offline, applying the resulting controller online becomes much faster than directly solving the optimization problem at each time step. In [45], the parameters of the feed-forward NN were trained to maximize the robustness of the STL specifications. Reinforcement learning algorithms such as deep Q-learning (DQN) and learning from demonstrations (Lfd) were used for learning the NN controller in [119] and [106], respectively. The works of [43, 44] used a Recurrent Neural Network (RNN) [120] instead of a feed-forward NN, which is more suitable for control with STL specifications since the satisfaction of an STL formula depends on a state trajectory (not just the current state). In [43, 44], RNN parameters

were trained using imitation learning and model-based reinforcement, respectively.

However, the afore-cited previous works can deal with *only one* STL specification (i.e., the NN controller is trained for only one prescribed STL specification), and did not explicitly deal with multiple STL specifications. In particular, it may be more desirable and flexible in practice that the user is able to choose an STL specification freely from multiple STL specifications, such as the case where a surveillance task in a certain region (building, park, *etc.*) changes from day to day. A naive approach to achieve this would be to learn a set of NN controllers independently for all the candidate STL specifications, although this would be expensive in terms of memory consumption and computational resources if the number of the STL specifications is large (i.e., the number of the parameters to be learned could increase as the number of the candidate STL specifications increases). Hence, we here consider learning a *single* NN controller, in which not only the states of the system but also a chosen STL specification are regarded as inputs to the NN. We in particular focus on learning an RNN controller since as previously mentioned, it is suitable for dealing with history-dependent nature of the STL specifications. The above formulation leads us to the following question to be answered, which has not been investigated in previous works of literature:

How should we encode the STL specifications by vectors, so that they are readable to the RNN?

Naively, we could assign any unique numbers or one-hot vectors to all the STL specifications and use them as the inputs to the RNN. Although these approaches are easy to implement, information about the similarities in the specifications cannot be encoded with these schemes. Thus, the underlying function to be learned can be unreasonably complex as the number of candidate STL specifications increases, which leads to increased computation time for training as well as a critical failure in control execution.

Motivated by the above issue, we propose a novel scheme for constructing a vector representation of STL specifications, called *STL2vec*, which captures similarities between the STL specifications. The vectors for the specifications will be trained based on Word2vec [121, 122], which is a technique widely known in natural language processing. Then, the parameters for the RNN are trained by taking the vectors obtained by STL2vec and state trajectory as the inputs (to the RNN) and defining a loss function by the average of negative robustness scores for all the candidate STL specifications.

The main contributions of this work are summarized as follows. First, we propose a method for obtaining vector representations of STL specifications that capture their similarities in terms of control policy. Although STL2vec is constructed based on the concept of Word2vec, which is a technique that maps a word of the natural language onto the vector space, how to generate a dataset to learn appropriate vector representations for STL specifications in control synthesis is not a trivial problem.

We address the details on how to construct dataset to train an appropriate STL2vec in Section 4.4. Second, by using the vectors generated by STL2vec, we train the controller that can deal with multiple STL specifications with one RNN model. Naively, if one aims to train the NN controller to deal with multiple specifications by simply applying existing NN-based control synthesis methods such as [43–45, 106, 119], one needs to ready NN models as many as the number of the STL specifications, which leads to large memory consumption and computational resources (for details, see subsection 4.4.3). Moreover, the proposed method has the potential to accelerate the training as will be seen in the case study of subsection 4.5. Although there exist other methods that make the NN controller flexible such as [43] and [123], which can handle (unknown) obstacles by using control barrier function, these methods are restricted to the specification of only collision avoidance and thus are not intended to deal with multiple temporal logic specifications. The proposed method in this study allows us to overcome such limitations; it allows the user to have a flexibility of selecting an STL specification freely from multiple ones, and this will be achieved by constructing STL2vec.

Chapter 2

Robust Learning-Based Iterative Model Predictive Control for Unknown Non-Linear Systems

This chapter presents a learning-based iterative model predictive control (MPC) scheme for unknown (Lipschitz continuous) non-linear dynamical systems. The proposed method begins by learning the unknown part of the controlled system using a Gaussian process (GP), which helps derive multi-step reachable sets that are guaranteed to encompass the actual system states. At each time step in each iteration, the MPC controller calculates a sequence of control inputs that robustly satisfy state and control constraints, as well as terminal constraints based on the GP-based reachable sets. Then only the first control input is applied to the system. After the iteration, we reset the initial state, and the same procedure is executed with the MPC optimization problem defined by the updated terminal set and cost. As iteration goes on, improvement of the control performance is expected since more data is obtained and the environment is progressively explored. The proposed method provides properties such as recursive feasibility and input to state stability of the goal region under certain assumptions. Moreover, bound on the performance cost in each iteration associated with the implementation of the proposed MPC scheme is also analyzed. The results of the simulation study show that the proposed control scheme can iteratively improve the control performance.

This chapter is organized based on “Robust Learning-Based Iterative Model Predictive Control for Unknown Non-Linear Systems” [124] which appeared in IET Control Theory & Applications.

Notations: A function $\alpha : [0, a) \rightarrow [0, \infty)$ ($a \in (0, \infty]$) is of class- $\mathcal{K}$ if it is continuous, strictly increasing, and satisfies $\alpha(0) = 0$. It is of class- $\mathcal{K}_\infty$ if, in addition, $a = \infty$ and $\alpha(r) \rightarrow \infty$ as $r \rightarrow \infty$. A function $\beta : [0, a) \times [0, \infty) \rightarrow [0, \infty)$ is of class- $\mathcal{KL}$ if for each fixed $t \geq 0$, $\beta(\cdot, t)$ is of class- $\mathcal{K}$,

and for each fixed $r \in [0, a)$, $\beta(r, \cdot)$ is non-increasing in t and satisfies $\beta(r, t) \rightarrow 0$ as $t \rightarrow \infty$. For a set $\mathcal{S} \subset \mathbb{R}^n$, the distance-to-set notation is $|x|_{\mathcal{S}} := \inf_{y \in \mathcal{S}} \|x - y\|$, where $\|\cdot\|$ denotes a chosen norm (typically the Euclidean norm).

2.1 Preliminaries

2.1.1 System description

We consider a nonlinear, discrete-time system of the form:

$$x_{t+1} = f(x_t, u_t) = \underbrace{h(x_t, u_t)}_{\text{known}} + \underbrace{g(x_t, u_t)}_{\text{unknown}},$$

$$x_t \in \mathcal{X}, u_t \in \mathcal{U}, \quad (2.1)$$

where x_t and u_t denote the state and control input at time $t \in \mathbb{N}$, respectively. We assume that $\mathcal{X}$ and $\mathcal{U}$ are given constraint sets and are compact (closed and bounded), with $0 \in \mathcal{X}$ and $0 \in \mathcal{U}$. The term $h : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ captures the known system dynamics, while the function $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is a function that accounts for the unknown part of the dynamics that are either difficult to model accurately or are unknown. Throughout this study, we use both notations $F(x, u)$ and $F(z)$ with $z = [x^\top, u^\top]^\top$ for any function F that takes a state $x \in \mathbb{R}^n$ and control $u \in \mathbb{R}^m$ as inputs, depending on the context. In addition, we assume that the functions h and g are Lipsitz continuous as the following.

Assumption 1. *The functions h and g are Lipsitz continuous, i.e., there exist constants $L_h > 0$ and $L_g > 0$ such that the following inequalities hold for all $z_1, z_2 \in \mathbb{R}^{n+m}$.*

$$|h(z_1) - h(z_2)| \leq L_h |z_1 - z_2|,$$

$$|g(z_1) - g(z_2)| \leq L_g |z_1 - z_2|.$$

2.1.2 Gaussian process regression

In this study, we utilize the Gaussian process regression (GPR) to estimate the unknown part of the dynamics g . We separately learn each element g_i using the measurement data $\mathcal{D}_i = \{Z, Y_i\}$, where $Z = [z_1, \dots, z_{N_{\text{data}}}]^\top$, $Y_i = [y_{1,i}, \dots, y_{N_{\text{data}},i}]^\top$, $z_t = [x_t^\top, u_t^\top]^\top$, and $y_{t,i} = g_i(x_t, u_t) = f_i(x_t, u_t) - h_i(x_t, u_t)$. In GPR, we first need to specify the prior mean function and the covariance function. In this study, we consider the zero prior mean function and use the following squared exponential covariance function:

$$k_i(z_p, z_q) = \sigma_i^2 \exp\left(-\frac{1}{2}(z_p - z_q)^\top \Lambda_i^{-1}(z_p - z_q)\right) + \delta_{pq} \sigma_\epsilon^2, \quad (2.2)$$

where $\delta_{pq}\sigma_\epsilon^2$ is a noise covariance function, $\Lambda_i := \text{diag}([\lambda_{1,i}, \dots, \lambda_{n+m,i}])$ and σ_i are the hyperparameters typically determined by solving evidence maximization problem [52]. Then, with the dataset $\mathcal{D}_i$, the posterior distribution at a point $z \in \mathbb{R}^{n+m}$ can be computed as a Gaussian distribution $\mathcal{N}(\mu_{g_i}(z), \sigma_{g_i}^2(z))$ with

$$\mu_{g_i}(z) = k_i^{*\top}(z) K_i^{-1} Y_i, \quad (2.3)$$

$$\sigma_{g_i}^2(z) = k_i(z, z) - k_i^{*\top}(z) K_i^{-1} k_i^*(z), \quad (2.4)$$

where $k_i^*(z) = [k_i(z, z_1), \dots, k_i(z, z_{N_{\text{data}}})]^\top \in \mathbb{R}^{N_{\text{data}}}$ and $K_i \in \mathbb{R}^{N_{\text{data}} \times N_{\text{data}}}$ is a kernel matrix whose each entry is defined as $[K_i]_{p,q} = k_i(z_p, z_q)$, $p, q \in \mathbb{N}_{1:N_{\text{data}}}$. The resulting multivariate GP approximation of the unknown function g is then given by stacking the individual dimensions, i.e., $\mu_g = [\mu_{g_1}, \dots, \mu_{g_n}]^\top$ and $\sigma_g = \text{diag}([\sigma_{g_1}, \dots, \sigma_{g_n}]^\top)$.

2.1.3 Propagation of the confidence regions

To incorporate the learned GPR model into the proposed predictive control strategy, we need a multi-step ahead prediction of the system evolution given a control input sequence. To this end, we here introduce a derivation proposed in [55]. Since it is hard to derive confidence regions for arbitrary given function g , we first restrict each element of the function g , ($g_i : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$) to the class of the reproducing kernel Hilbert space (RKHS), which is one of the common assumptions in the GPR-based control methods [125, 126].

Assumption 2. For each $i \in \mathbb{N}_{1:n}$, the function g_i lies in the RKHS corresponding to a given kernel function $k_i : \mathbb{R}^{n+m} \times \mathbb{R}^{n+m} \rightarrow \mathbb{R}$ with the induced norm denoted as $\|\cdot\|_{k_i}$. Moreover, $\|g_i\|_{k_i} \leq b_i$ holds for all $i \in \mathbb{N}_{1:n}$ and given $b_i > 0$. $\square$

Under Assumption 2, the following lemma can be obtained [59], which is used to derive the confidence regions.

Lemma 1. Given the training dataset $\mathcal{D}_i$ and let Assumption 2 hold. Then, the following holds for all $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $i \in \mathbb{N}_{1:n}$:

$$g_i(x, u) \in [\mu_{g_i}(z) \pm \beta_i \sigma_{g_i}(z)], \quad (2.5)$$

where $z = [x^\top u^\top]^\top$, $\beta_i = \sqrt{b_i^2 - Y_i^\top K_i^{-1} Y_i}$. $\square$

For the proof of Lemma 1, see Proposition 1 in [59]. We shall then derive a multi-step ahead propagation of ellipsoidal confidence regions of system states given a sequence of control inputs. We

first introduce the following notation to express an ellipsoid:

$$\mathcal{E}(p, P) := \{x \in \mathbb{R}^n | (x - p)^\top P^{-1} (x - p) \leq 1\}, \quad (2.6)$$

where $p \in \mathbb{R}^n$ is a center and P is a symmetric positive definite matrix that encodes the shape of an ellipsoid. Let an ellipsoid $\Phi_t = \mathcal{E}(p_t, P_t)$ be the confidence region of the system states at time t . Then, we derive an ellipsoid at the next time step $\Phi_{t+1} = \mathcal{E}(p_{t+1}, P_{t+1})$ that includes all possible states evolved from any states within Φ_t with a control input $u_t \in \mathcal{U}$. To this end, we first consider the linear approximation of the mean prediction $h(z) + \mu_g(z)$ around a point $\bar{z}_t = (p_t, u_t)$ as follows:

$$f(z) \approx h(\bar{z}_t) + J_h(\bar{z}_t)(z - \bar{z}_t) + P_{\mu_g}^{\bar{z}_t}(z) = \tilde{f}(z), \quad (2.7)$$

where $J_h(\bar{z}_t) = [A_{h,t}, B_{h,t}]$ is the Jacobian of the function h at $\bar{z}_t$, and $P_{\mu_g}^{\bar{z}_t}(z) = \mu_g(\bar{z}_t) + J_{\mu_g}(\bar{z}_t)(z - \bar{z}_t)$, where $J_{\mu_g}(\bar{z}_t) = [A_{\mu_g,t}, B_{\mu_g,t}]$ is the Jacobian of μ_g at $\bar{z}_t$. Then, by applying the Lagrangian remainder theorem on the linearization of h and using the inequality $|g_j(z) - P_{\mu_{g,j}}^{\bar{z}_t}(z)| \leq |g_j(z) - \mu_{g_j}(z)| + |P_{\mu_{g,j}}^{\bar{z}_t}(z) - \mu_{g_j}(z)|$, we have the following for all $z \in \mathcal{X} \times \mathcal{U}$ and $1 \leq j \leq n$:

$$|f_j(z) - \tilde{f}_j(z)| \leq \frac{L_{\nabla h_j}}{2} \|z - \bar{z}_t\|_2^2 + |P_{\mu_{g,j}}^{\bar{z}_t}(z) - \mu_{g_j}(z)| + |\mu_{g_j}(z) - g_j(z)|, \quad (2.8)$$

where $L_{\nabla h_j}$ is the Lipsitz constant of the gradient ∇h_j and the second and third terms in the right-hand side of (2.8) are further upper-bounded by

$$|P_{\mu_{g,j}}^{\bar{z}_t}(z) - \mu_{g_j}(z)| \leq \frac{L_{\nabla \mu_{g_j}}}{2} \|z - \bar{z}_t\|_2^2, \quad (2.9)$$

$$|\mu_{g_j}(z) - g_j(z)| \leq \beta(\sigma_{g_j}(\bar{z}_t) + L_{\sigma_{g_j}} \|z - \bar{z}_t\|_2), \quad (2.10)$$

where L_{σ_j} and $L_{\nabla \mu_{g_j}}$ are the Lipschitz constants of σ_{g_j} and $\nabla \mu_{g_j}$. Moreover, in (2.10), we used Lemma 1.

Then, we can interpret (2.8) as the edges of the confidence hyper-rectangle:

$$\tilde{m}(z) = \tilde{f}(z) \pm \left[\beta(\sigma_g(\bar{z}_t) + L_{\sigma} \|z - \bar{z}_t\|_2) + \frac{L_{\nabla h} + L_{\nabla \mu_g}}{2} \|z - \bar{z}_t\|_2^2 \right], \quad (2.11)$$

where $\tilde{f} = [\tilde{f}_1, \dots, \tilde{f}_n]$, $L_{\nabla h} = [L_{\nabla h_1}, \dots, L_{\nabla h_n}]$ and $L_{\nabla \mu_g} = [L_{\nabla \mu_{g_1}}, \dots, L_{\nabla \mu_{g_n}}]$. Now we consider propagating the ellipsoidal region Φ_t through the confidence region of $f(z)$ and deriving Φ_{t+1} covering that region. First, by using the well-known linear transformation of the ellipsoidal region [127] and allowing a slight abuse of notation, we can write the propagation of the ellipsoid Φ_t through the linear function (2.7) as follows:

$$\tilde{f}(\Phi_t, u_t) = \mathcal{E}(h(\bar{z}_t) + \mu_g(\bar{z}_t), (A_{h,t} + A_{\mu_{g,t}})P(A_{h,t} + A_{\mu_{g,t}})^\top). \quad (2.12)$$

We can see that the resulting region is still ellipsoid. Next, we define the upper bound of $\|z - \bar{z}_t\|_2$ as

$$I(\Phi_t, u_t) := \max_{z(x)=(x, u_t), x \in \Phi_t} \|z(x) - \bar{z}_t\|_2. \quad (2.13)$$

Then we have the upper bound of the remaining term in (2.11) as

$$\begin{aligned} \tilde{d}(\Phi_t, u_t) &= \beta(\sigma_g(\bar{z}_t) + L_\sigma I(\Phi_t, u_t)) \\ &\quad + (L_{\nabla h} + L_{\nabla \mu_g})I^2(\Phi_t, u_t)/2. \end{aligned} \quad (2.14)$$

From $\|z(x) - \bar{z}_t\|_2 = \|x - p_t\|_2$ and the property of the ellipsoid [127], we can calculate $I(\Phi_t, u_t)$ by the maximum eigenvalue of P^{-1} . Then we over-approximate $\tilde{d}(\Phi_t, u_t)$ with an ellipsoid as follows:

$$0 \pm \tilde{d}(\Phi_t, u_t) \subset \mathcal{E}(0, P_{\tilde{d}}(\Phi_t, u_t)), \quad (2.15)$$

where we can obtain $P_{\tilde{d}}(\Phi_t, u_t)$ by using the property of the ellipsoid: $a \pm b := [a_1 \pm b_1] \times \dots \times [a_n \pm b_n] \subset \mathcal{E}(a, \sqrt{n} \text{diag}([b_1, \dots, b_n]))$, $\forall a, b \in \mathbb{R}^n$. By combining the results above, we can compute an over-approximation of the propagation as the following:

$$\Phi_{t+1}^+ = \tilde{f}(\Phi_t, u_t) \oplus \mathcal{E}(0, P_{\tilde{d}}(\Phi_t, u_t)). \quad (2.16)$$

Although the Minkowski sum Φ_{t+1}^+ is not an ellipsoid in general, we can over-approximate this set with an ellipsoid by using the following property regarding the Minkowski sum of two ellipsoidal sets:

$$\mathcal{E}(p_1, P_1) \oplus \mathcal{E}(p_2, P_2) \subset \mathcal{E}_c(\tilde{p}, \tilde{P}), \quad (2.17)$$

where $\tilde{p} = p_1 + p_2$, $\tilde{P} = (1 + c^{-1})P_1 + (1 + c)P_2$, for all $c > 0$. Thus, from (2.12), (2.15), (2.16), and (2.17), the evolution of the confidence ellipsoids is obtained as

$$\Phi_{t+1} = \mathcal{E}(p_{t+1}, P_{t+1}) := \tilde{m}(\Phi_t, u_t), \quad (2.18)$$

with $p_{t+1} = h(\bar{z}_t) + \mu_g(\bar{z}_t)$ and $P_{t+1} = (1 + c^{-1})(A_{h,t} + A_{\mu_g,t})P(A_{h,t} + A_{\mu_g,t})^\top + (1 + c)P_{\tilde{d}}(\Phi_t, u_t)$. The multi-step ahead prediction of the confidence regions can be obtained by iteratively applying $\tilde{m}$. We note here that the evolution of the center of ellipsoids is equivalent to that of the certainty equivalent model (prediction with mean function) from the definition above. For the ellipsoidal confidence region defined above, we have the following properties (i.e., guarantee on the uncertainty propagation and monotonicity property) that are important in the proposed control strategy.

Lemma 2. *For the propagation of confidence region (2.18), the following two properties hold. (i) Suppose x_0 is included in an initial ellipsoid Φ_0 . Then, for any control inputs $u_0, \dots, u_{T-1} \in \mathcal{U}$, the resulting sequence of the states $x_{0:T}$ based on (2.1) and the confidence ellipsoids $\Phi_{0:T}$ with (2.18) meet*

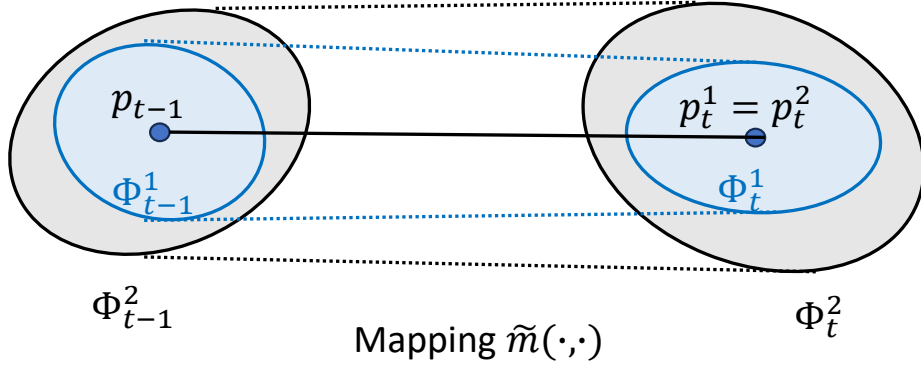


Figure 2.1: The monotonicity property of the mapping $\tilde{m}$. If $\Phi_{t-1}^1 \subseteq \Phi_{t-1}^2$ holds for two ellipsoids $\Phi_{t-1}^1 = \mathcal{E}(p_{t-1}, P_{t-1}^1)$ and $\Phi_{t-1}^2 = \mathcal{E}(p_{t-1}, P_{t-1}^2)$, the ellipsoids mapped by $\tilde{m}$ with any control input $u_{t-1} \in \mathcal{U}$ hold the relationships $\Phi_t^1 \subseteq \Phi_t^2$ and $p_t^1 = p_t^2$.

$x_t \in \Phi_t, \forall t \in \mathbb{N}_{0:T}$. (ii) if two ellipsoids $\Phi_{t-1}^1 := \mathcal{E}(p_{t-1}, P_{t-1}^1)$ and $\Phi_{t-1}^2 := \mathcal{E}(p_{t-1}, P_{t-1}^2)$ have the relationship $\Phi_{t-1}^1 \subseteq \Phi_{t-1}^2$, then the ellipsoids at the next time step $\Phi_t^1 = \tilde{m}(\Phi_{t-1}^1, u_{t-1}) := \mathcal{E}(p_t^1, P_t^1)$ and $\Phi_t^2 = \tilde{m}(\Phi_{t-1}^2, u_{t-1}) := \mathcal{E}(p_t^2, P_t^2)$ satisfy $\Phi_t^1 \subseteq \Phi_t^2$ and $p_t^1 = p_t^2$ for any control input $u_{t-1} \in \mathcal{U}$.

Proof. The first property holds from the construction of the confidence regions. In the following, we show the second property. From the definition of the propagation of the ellipsoidal confidence region, the center of the ellipsoid p_t is computed by $h(\bar{z}_{t-1}) + \mu(\bar{z}_{t-1})$ with $\bar{z}_{t-1} = [p_{t-1}, u_{t-1}]^\top$. Thus, $p_t^1 = p_t^2 = h(\bar{z}_{t-1}) + \mu(\bar{z}_{t-1})$. Moreover, from (2.12), (2.16) and (2.17), we can easily see that $\Phi_t^1 \subseteq \Phi_t^2$ holds if $\Phi_{t-1}^1 \subseteq \Phi_{t-1}^2$. $\square$

We call the second property as *monotonicity property* and the illustrative explanation is shown in Fig. 2.1, which is important in the proposed control scheme. Although the discussion in this subsection is restricted to the specific derivation proposed in [55], we here emphasize that any confidence sets that satisfy the properties in Lemma 2 can be used in the proposed control scheme. For example, the uncertainty propagation technique developed in [63] can also be used in the proposed scheme.

2.2 Problem Statement

We first consider the following optimal control problem that requires satisfying the state and control constraints at all time steps and the system states reaching within the goal set $\mathcal{O}$ at some point in time

while minimizing the performance cost:

$$J_{0 \rightarrow \infty}^*(x_s) = \min_{u_0, u_1, \dots} \sum_{k=0}^{\infty} \ell(x_k, u_k), \quad (2.19a)$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k), \quad \forall k \geq 0, \quad (2.19b)$$

$$x_0 = x_s, \quad (2.19c)$$

$$x_k \in \mathcal{X}, \quad \forall k \in \{1, 2, \dots\}, \quad (2.19d)$$

$$u_k \in \mathcal{U}, \quad \forall k \in \{0, 1, \dots\}, \quad (2.19e)$$

$$x_{k'} \in \mathcal{O} \quad \exists k' \in \{1, 2, \dots\}, \quad (2.19f)$$

where $x_s \in \mathcal{X}$ is the initial state and the function $\ell : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ represents the stage cost of the problem. We make the following assumptions on the function ℓ .

Assumption 3. $\ell(x, u) = 0, \forall x \in \mathcal{O}, \forall u \in \mathcal{U}$, where $\mathcal{O}$ is control invariant set, i.e., $x \in \mathcal{O} \implies \exists u \in \mathcal{U}, f(x, u) \in \mathcal{O}$. Moreover, we assume that the control policy $\pi_{\mathcal{O}} : \mathcal{O} \rightarrow \mathcal{U}$ that makes $\mathcal{O}$ forward invariant exists and is given.

Assumption 4. The function $\ell(\cdot, \cdot)$ is continuous and jointly convex in its arguments. Moreover, we assume that for all $x \in \mathbb{R}^n$, and for all $u \in \mathbb{R}^m$, there exist $\alpha_x^l, \alpha_x^u, \alpha_u^l$ and $\alpha_u^u \in \mathcal{K}_{\infty}$ such that:

$$\begin{aligned} \alpha_x^l(|x|_{\mathcal{O}}) &\leq \ell(x, 0) \leq \alpha_x^u(|x|_{\mathcal{O}}), \\ \alpha_u^l(|u|_{\mathcal{U}_{\mathcal{O}}}) &\leq \ell(0, u) \leq \alpha_u^u(|u|_{\mathcal{U}_{\mathcal{O}}}), \end{aligned}$$

where $\mathcal{U}_{\mathcal{O}} = \{u \in \mathcal{U} \mid \exists x \in \mathcal{O}, f(x, u) \in \mathcal{O}\}$.

We here also define the Input-to-State Stability (ISS) for the invariant set $\mathcal{O}$ and given GP model, which is used in the theoretical analysis of subsection 2.4.

Definition 1. Given a control policy $\pi(\cdot)$ and trained GP model, the closed-loop system is ISS with respect to the invariant set $\mathcal{O}$ and given GP model if for all errors between the mean prediction and the actual system states $e_t = \mu_g(x_{t-1}, u_{t-1}) - g(x_{t-1}, u_{t-1})$, $u_{t-1} = \pi(x_{t-1}) \in \mathcal{U}$, $t \geq 0$, and $x_0 \in \mathcal{X}$, there exist a class- $\mathcal{KL}$ function $\zeta(\cdot, \cdot)$ and a class- $\mathcal{K}$ function $\gamma(\cdot)$ (see the notation part in the beginning of this chapter for definitions) such that:

$$|x_{t+1}|_{\mathcal{O}} \leq \zeta(|x_0|_{\mathcal{O}}, t+1) + \gamma(\sup_{k \in \{0, \dots, t\}} \|e_k\|). \quad (2.20)$$

In this study, we consider approaching to (2.19) by performing MPC with the learned dynamics discussed in Section 2.1. Since an MPC optimization problem solved at each time step has a finite and short horizon window, the control input sequence obtained by solving the optimization problem

in a receding horizon manner is generally a sub-optimal solution to the problem (2.19). Moreover, a designer needs to appropriately construct a terminal set and cost to ensure the feasibility of the optimization problem for all time steps, which is especially challenging in our setting because of the unknown part of the dynamics in (2.1). To tackle these challenges, we consider iteratively performing the MPC and improving the performance cost by effectively updating the terminal components of the MPC problem with the data collected in the past iterations. To this end, we first assume that at least one sequence of control inputs (which may be far from the optimal solution of (2.19)) that steer the confidence region of the system state calculated by the current GP prediction into the control invariant set $\mathcal{O}$ and the associated confidence regions are given before the execution as summarized in the following assumption.

Assumption 5. A sequence of control inputs $u_{0|0}^0, u_{1|0}^0, \dots, u_{T_0-1|0}^0 \in \mathcal{U}$ and corresponding multi-step confidence regions of the system (2.1) $\Phi_{1|0}^0, \Phi_{2|0}^0, \dots, \Phi_{T_0|0}^0 \subseteq \mathcal{X}$ with $\Phi_{0|0}^0 = x_s$, $\Phi_{t+1|0}^0 \supseteq \tilde{m}(\Phi_{t|0}^0, u_{t|0}^0)$, and $\Phi_{T_0|0}^0 \subseteq \mathcal{O}$ are given in advance.

Subscripts for control inputs and corresponding ellipsoidal sets in Assumption 5 are adapted from the discussion in subsection 2.3.1. We also denote the control inputs generated by the proposed MPC scheme at time $t \in \{0, 1, \dots\}$ in iteration $j \in \{1, 2, \dots\}$ and the resulting actual system states observed at time $t \in \{0, 1, \dots\}$ in iteration $j \in \{1, 2, \dots\}$ as u_t^j and x_t^j .

Since practically it is hard to obtain the global optimal solution of the problem (2.19) in our setting, we propose an iterative MPC strategy that alternatively satisfies the following properties at each iteration j : **(P1)** With the proposed MPC strategy, constraints in (2.19) are robustly satisfied despite the model uncertainty **(P2)** The proposed MPC optimization problem is feasible for all time $t > 0$ **(P3)** The resulting closed-loop system is ISS for the goal set $\mathcal{O}$ with certain assumptions **(P4)** There always exists a control input sequence that safely steers the state of the system (2.1) in the set $\mathcal{O}$ for all time during the control execution **(P5)** With the certain restrictions on the proposed scheme, the system states converge to the set $\mathcal{O}$ in finite time with the proposed MPC scheme **(P6)** The iteration cost defined as $J_{0 \rightarrow T^j}^j(x_s) := \sum_{t=0}^{T^j} \ell(x_t^j, u_t^j)$ is upper bounded as $J_{0 \rightarrow T^j}^j(x_s) \leq Q^{j-1}(x_s) + L \sum_{t=0}^{T^j} \|e_t^j\|$, where the function Q^j is non-increasing with respect to the number of iteration j , $L > 0$, e_t^j is the error between the (one-step) certainty equivalent prediction and actual state, and T^j is execution time at iteration j . The concrete definition of Q^j and $L > 0$ is explained in Section 2.3.

In the following, we first introduce the proposed MPC strategy in Section 2.3. Then, we show that the proposed control scheme meets the properties **(P1)**–**(P6)** in Section 2.4. The property **(P1)** is ensured from the definition of the proposed MPC formulation and the properties **(P2)**–**(P6)** are shown in Theorem 1, Theorem 2, Remark 1, Remark 2, and Theorem 3 respectively.

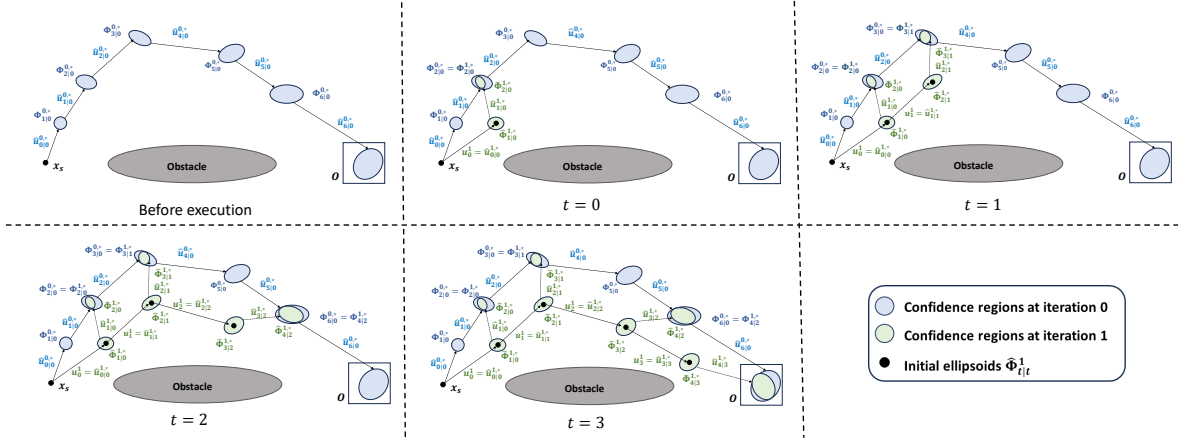


Figure 2.2: An example of the execution in iteration 1 when the initial control sequence and the corresponding sequence of confidence regions (blue ellipsoids) are given by $\hat{u}_{0:6|0}^{0,*}$ and $\Phi_{1:6|0}^{0,*}$ as shown in the left above figure. The horizon length is set to $N = 2$ in this example. At time 0 in iteration 1 (middle above figure), the LMPC optimization (2.21) at $x_0^j = x_s$ is solved, and the optimal sequence of inputs $\hat{u}_{0|0}^{1,*}, \hat{u}_{1|0}^{1,*}$ and the corresponding confidence regions $\Phi_{1|0}^{1,*}, \Phi_{2|0}^{1,*}$ are obtained to satisfy the terminal constraint (in this case, $\Phi_{2|0}^{1,*} \subseteq \Phi_{2|0}^{0,*}$). Then, the first control input $u_0^1 = \hat{u}_{0|0}^{1,*}$ is applied to the system and the next state x_1^1 is observed. This process is repeated until the prescribed time T^1 is reached. After iteration 1, the terminal set $\mathcal{SS}^0$ and cost Q^0 are updated by following the procedures in subsections 2.3.1, 2.3.2 and 2.3.3. The procedures in the later iterations similarly follow.

2.3 Proposed Method

In this section, we explain the proposed method to solve the problem discussed in Section 2.2. We here assume that the GP model is trained with an initially given dataset and is not updated during the execution. The control scheme including GP update is additionally discussed later in subsection 2.5.1. As discussed in Section 2.2, we consider iteratively performing MPC as gathering data under Assumption 5. The entire control scheme is summarized in Algorithm 1 and Fig. 2.2. The proposed LMPC optimization problem solved at time t in iteration j is defined as follows with $\hat{p}_{0|t-1}^{j,*} = x_0^j = x_s$:

$$J_{t \rightarrow t+N_t^j}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*}) = \min_{\hat{u}_{t|t}, \dots, \hat{u}_{t+N_t^j-1|t}} \left[\sum_{k=t}^{t+N_t^j-1} \ell(\hat{x}_{k|t}^j, \hat{u}_{k|t}^j) + Q^{j-1}(\hat{\Phi}_{t+N_t^j|t}^j) \right] \quad (2.21a)$$

s.t.

$$\hat{x}_{k+1|t}^j = \hat{f}(\hat{x}_{k|t}^j, \hat{u}_{k|t}^j), \quad \hat{x}_{t|t}^j = x_t^j, \quad (2.21b)$$

$$\hat{\Phi}_{k+1|t}^j = \mathcal{E}(\hat{p}_{k+1|t}^j, \hat{p}_{k+1|t}^j) = \tilde{m}(\hat{\Phi}_{k|t}^j, \hat{u}_{k|t}^j), \quad (2.21c)$$

$$\hat{\Phi}_{t|t}^j = \mathcal{B}(\hat{p}_{t|t-1}^{j,*}, |x_t^j - \hat{p}_{t|t-1}^{j,*}|), \quad (2.21d)$$

$$\hat{\Phi}_{k|t}^j \subseteq \mathcal{X}, \quad \forall k \in \mathbb{N}_{t:t+N_t^j}, \quad (2.21e)$$

$$\hat{u}_{k|t}^j \in \mathcal{U}, \quad \forall k \in \mathbb{N}_{t:t+N_t^j-1}, \quad (2.21f)$$

$$\hat{\Phi}_{t+N_t^j|t}^j \subseteq \Phi = \mathcal{E}(p, P), \quad \hat{p}_{t+N_t^j|t}^j = p, \exists \Phi \in \mathcal{SS}^{j-1}, \quad (2.21g)$$

where the function $\hat{f}$ is the mean prediction of the dynamics (2.1), i.e., $\hat{f}(\hat{x}_{k|t}^j, \hat{u}_{k|t}^j) = h(\hat{x}_{k|t}^j, \hat{u}_{k|t}^j) + \mu_g(\hat{x}_{k|t}^j, \hat{u}_{k|t}^j)$. Moreover, $N_t^j > 0$ is the horizon length at time t in iteration j . If the set $\mathcal{O}$ is selected as the terminal set at the previous time step $t' (< t)$, N_t^j is defined as $N_t^j = \max(N - (t' - t), 0)$ and if not, $N_t^j = N$, where N is the maximum number of the horizon length chosen by the designer. If $N_t^j = 0$ (i.e., initial ellipsoid is already within the set $\mathcal{O}$), the cost for MPC (2.21a) is defined as $J_{t \rightarrow t+N_t^j}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*}) = 0$. $\hat{p}_{t|t-1}^{j,*}$ in (2.21d) is a center of the optimal ellipsoid at time t obtained at previous time $t-1$ by solving (2.21) and $\mathcal{B}(p, r)$ in (2.21d) is a ball with center p and radius r . The reason we define the initial confidence region in this way instead of just defining $\hat{\Phi}_{t|t}^j = x_t^j$ when $t \neq 0$ is to ensure the recursive feasibility as explained later in subsection 2.4.1. $\mathcal{SS}^{j-1}$ and Q^{j-1} are the collection of the terminal sets and the terminal cost for iteration j , respectively. The set $\mathcal{SS}^{j-1}$ basically consists of the confidence regions calculated in the previous iterations, and the cost Q^{j-1} is defined based on the components within $\mathcal{SS}^{j-1}$. The details regarding how to construct these terminal components are provided in the following subsections. The constraint (2.21e) is the state constraint that enforces the actual system (2.1) robustly satisfies the original state constraint (2.19d) despite the model uncertainty, and the constraint (2.21g) is the terminal constraint that enforces the last confidence region $\hat{\Phi}_{t+N_t^j|t}^j$ to be included in a component within $\mathcal{SS}^{j-1}$ and the centers of these ellipsoids are matched. The inclusion condition in (2.21g) can be rewritten as $P - \hat{P}_{t+N_t^j|t}^j \succeq 0$. We denote the optimal control sequence, certainty equivalent predictions, and the corresponding confidence regions

Algorithm 1: Control execution

Input : $\hat{f}, \tilde{m}$ (GP prediction models); x_s (initial state); $\mathcal{SS}^0$ (initial terminal set); N_{ite} (number of iteration); T^j (execution time steps in each iteration)

```

1 ; for  $j = 1, 2, \dots, N_{\text{ite}}$  do
2   Set the initial state by  $\hat{\Phi}_{0|0}^j = x_0^j = x_s$ ;
3   for  $t = 0, 1, \dots, T^j$  do
4     Solve problem (2.21) and obtain the solutions (2.22)–(2.24);
5     Apply the control input (2.25) to the system (2.1) and observe the next state  $x_{t+1}^j$ ;
6   end
7   Update the terminal set  $\mathcal{SS}^{j-1} \rightarrow \mathcal{SS}^j$  by following the procedures in subsections 2.3.1 and 2.3.2;
8   Update the terminal cost  $Q^{j-1}(\cdot) \rightarrow Q^j(\cdot)$  by following the procedures in subsection 2.3.3;
9 end

```

at time t in iteration j as

$$\hat{u}_{t:t+N_t^j-1|t}^{j,*} = [\hat{u}_{t|t}^{j,*}, \dots, \hat{u}_{t+N_t^j-1|t}^{j,*}], \quad (2.22)$$

$$\hat{x}_{t+1:t+N_t^j|t}^{j,*} = [\hat{x}_{t+1|t}^{j,*}, \dots, \hat{x}_{t+N_t^j|t}^{j,*}], \quad (2.23)$$

$$\hat{\Phi}_{t+1:t+N_t^j|t}^{j,*} = [\hat{\Phi}_{t+1|t}^{j,*}, \dots, \hat{\Phi}_{t+N_t^j|t}^{j,*}]. \quad (2.24)$$

Then, at each time t in iteration j , the following control input is applied to the system (2.1):

$$u_t^j = \begin{cases} \hat{u}_{t|t}^{j,*} & \text{if } N_t^j > 0, \\ \pi_{\mathcal{O}}(x_t^j) & \text{if } N_t^j = 0, \end{cases} \quad (2.25)$$

where $\pi_{\mathcal{O}}(\cdot)$ is the control policy that makes the set $\mathcal{O}$ forward invariant defined in Assumption 4. After applying the control input u_t^j , the state at the next time step x_{t+1}^j is observed. This procedure is iteratively executed until the prespecified execution time T^j is reached. After the iteration j , the terminal components Q^{j-1} and $\mathcal{SS}^{j-1}$ are updated with the data collected in iteration j by the procedures explained later in subsections 2.3.1, 2.3.2, and 2.3.3, and the iteration $j + 1$ is executed from the initial state $x_0^{j+1} = \hat{\Phi}_{0|0}^{j+1} = x_s$. In the following subsections, we will discuss the concrete procedures of constructing the set $\mathcal{SS}^j$ and the terminal cost Q^j .

2.3.1 Collection of terminal sets $\mathcal{SS}^j$

Let us explain how to construct the collection of the terminal sets $\mathcal{SS}^j$ based on the data collected in the past iterations. Specifically, the set $\mathcal{SS}^j$ is constructed to be a collection of subsets of the reachable set to the goal region $\mathcal{O}$ for the current GP model. First of all, based on Assumption

5, we define the initial collection of the terminal sets which is used in the execution in iteration 1 as $\mathcal{SS}^0 = \{\Phi_{1|0}^0, \Phi_{2|0}^0, \dots, \Phi_{T^0-1|0}^0, O\}$ with $\tilde{m}(\Phi_{T^0-1|0}^0, u_{T^0-1|0}^0) \subseteq O$. Here, we can easily see that any set within $\mathcal{SS}^0$ is a subset of the reachable set to O from Assumption 5. Then, at each time t in iteration 1, we solve the optimization (2.21) and obtain the optimal sequence of inputs $\hat{u}_{t:t+N_t^1-1|t}^{1,*}$ and the corresponding confidence regions $\hat{\Phi}_{t+1:t+N_t^1|t}^{1,*}$ with $\hat{\Phi}_{t+N_t^1|t}^{1,*} \subseteq \Phi_{t^*|0}^0, \exists t^* \in \mathbb{N}_{1:T^0-1}$ or $\hat{\Phi}_{t+N_t^1|t}^{1,*} \subseteq O$. Since it is hard to directly use these sets as terminal sets in future iterations, as discussed in the following subsection, we modify these sets with the procedure explained in the following subsection and construct the updated sets $\Phi_{t+1:t+N_t^1-1|t}^{1,*}$. For notational convenience, we define the sequence of ellipsoids $\Phi_{t+1|t}^{1,*}, \dots, \Phi_{t+N_t^1-1|t}^{1,*}, \Phi_{t^*|0}^0, \dots, \Phi_{T^0-1|0}^0$ as $\Phi_{t+1:t+T_t^1|t}^1 = \Phi_{t+1|t}^1, \dots, \Phi_{t+N_t^1-1|t}^1, \Phi_{t+N_t^1|t}^1, \dots, \Phi_{t+T_t^1|t}^1$. Similarly, we also define the sequence of the control inputs $\hat{u}_{t|t}^{1,*}, \dots, \hat{u}_{t+N_t^1-1|t}^{1,*}, u_{t^*|0}^0, \dots, u_{T^0-1|0}^0$ as $u_{t:t+T_t^1|t}^1 = u_{t|t}^1, \dots, u_{t+N_t^1-1|t}^1, u_{t+N_t^1|t}^1, \dots, u_{t+T_t^1|t}^1$. After the execution of iteration 1, we set the updated terminal set $\mathcal{SS}^1$ by appending the newly obtained sets during the execution of iteration 1, $\Phi_{t+1:t+N_t^1-1|t}^{1,*} = \Phi_{t+1:t+N_t^1-1|t}^1, \forall t \in \mathbb{N}_{0:T^1}$ to $\mathcal{SS}^0$. In the same way, at each time t in iteration j , $\Phi_{t+1:t+N_t^j|t}^j$ is computed and after the execution of iteration j , the set $\mathcal{SS}^j$ is constructed by appending the newly obtained sets at iteration j , $\Phi_{t+1:t+N_t^j-1|t}^{j,*} = \Phi_{t+1:t+N_t^j-1|t}^j, \forall t \in \mathbb{N}_{0:T^j}$ to $\mathcal{SS}^{j-1}$.

2.3.2 Backward enlargement of reachable sets

Practically, the forward reachable sets obtained at time t in iteration j , i.e., $\hat{\Phi}_{t+1:t+N_t^j|t}^{j,*}$ with $\hat{\Phi}_{t+k|t}^{j,*} = \mathcal{E}(\hat{p}_{t+k|t}^{j,*}, \hat{P}_{t+k|t}^{j,*})$ are likely to be too small to use as the terminal sets in the following iterations since the size of the uncertainty ellipsoids grows along with the time steps and the terminal predicted ellipsoid needs to be included in a terminal set in $\mathcal{SS}^{j-1}$, which leads to poor performance due to a small number of the available terminal sets. Since the obtained predictive sequence of the ellipsoidal set usually meets the terminal constraint with some margin, we can use this degree of freedom to enlarge the sets without breaking the inclusion relationships. To this end, we introduce a technique, Backward Enlargement (BE), which enlarges the reachable sets backward in time (see also Fig. 2.3). We denote the updated reachable sets calculated by BE as $\Phi_{t+k|t}^{j,*} = \mathcal{E}(p_{t+k|t}^{j,*}, P_{t+k|t}^{j,*}), \forall k \in \mathbb{N}_{1:N_t^j}$ with $\Phi_{t+N_t^j|t}^{j,*} = \Phi_{t^*+k^*|t^*}^{j,*} \supseteq \hat{\Phi}_{t+N_t^j|t}^{j,*}$, where $\Phi_{t^*+k^*|t^*}^{j,*} \in \mathcal{SS}^{j-1}$ is the terminal set selected at time t in iteration j . These sets are obtained by solving the following optimization problem from time

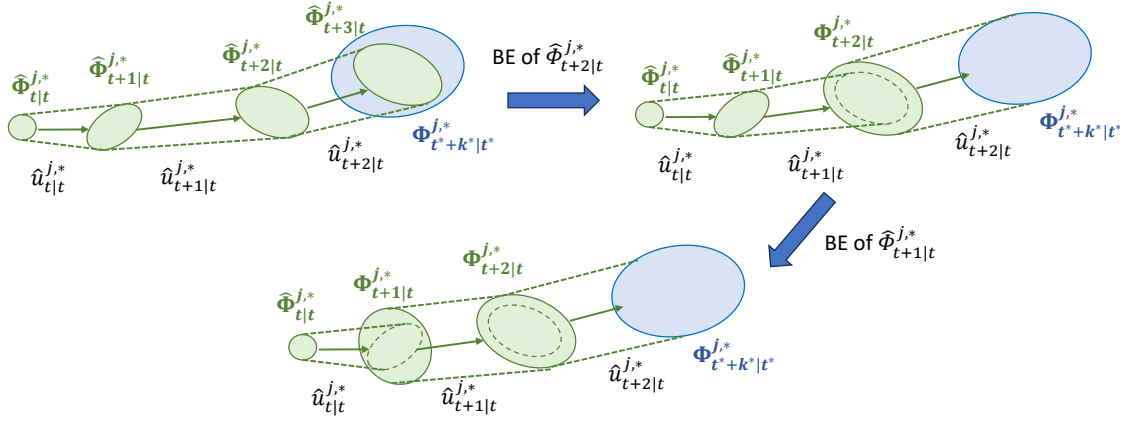


Figure 2.3: Backward Enlargement of the reachable sets. In this example, we first assume that the optimal control inputs $\hat{u}_{t:t+2|t}^{j,*}$ and reachable sets $\hat{\Phi}_{t:t+3|t}^{j,*}$ with $\hat{\Phi}_{t+3|t}^{j,*} \subseteq \Phi_{t^*+k^*|t^*}^{i*}$ are obtained by solving the problem (2.21) as shown in the left above figure. Then, the enlargement of the sets $\hat{\Phi}_{t+2|t}^{j,*}$ and $\hat{\Phi}_{t+1|t}^{j,*}$ are performed in this order by solving (2.26) with $\Phi_{t+3|t}^{j,*} = \Phi_{t^*+k^*|t^*}^{i*}$. With this scheme, we can enlarge the sets without breaking the inclusion relationships $\tilde{m}(\Phi_{t+k|t}^{j,*}, \hat{u}_{t+k|t}^{j,*}) \subseteq \Phi_{t+k+1|t}^{j,*}, \forall k \in \{1, 2\}$ by effectively using the fact that the terminal ellipsoid $\hat{\Phi}_{t+3|t}^{j,*}$ satisfies the terminal constraint (2.21g) with some margin.

$k = N_t^j - 1$ to 1:

$$\Phi_{t+k|t}^{j,*} (= \operatorname{argmax}_L \operatorname{Vol}(\Phi)) = \operatorname{argmax}_{L_P} \operatorname{tr}(L_P), \quad (2.26a)$$

$$\text{s.t. } \Phi = \mathcal{E}(\hat{p}_{t+k|t}^{j,*}, P), P = L_P L_P^\top, \Phi \subseteq \mathcal{X}, \quad (2.26b)$$

$$\mathcal{E}(p', P') := \tilde{m}(\Phi, \hat{u}_{t+k|t}^{j,*}), \quad (2.26c)$$

$$P - \hat{P}_{t+k|t}^{j,*} \succeq 0, P_{t+k+1|t}^{j,*} - P' \succeq 0, \quad (2.26d)$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix, L_P is a lower triangular matrix, $\operatorname{Vol}(\cdot)$ represents the volume of an ellipsoid, and we used the Cholesky decomposition in (2.26b) to avoid numerical instability. The condition (2.26d) enforces the new ellipsoids $\Phi_{t+k|t}^{j,*}$ include the old ones $\hat{\Phi}_{t+k|t}^{j,*}$ and the one step forwarded ellipsoid $\tilde{m}(\Phi_{t+k|t}^{j,*}, \hat{u}_{t+k|t}^{j,*})$ is included in the ellipsoid at the next time step $\Phi_{t+k+1|t}^{j,*}$ which is previously calculated by (2.26). This procedure is performed for all sequences of ellipsoids $\hat{\Phi}_{t+1:t+N_t^j|t}^{j,*}, \forall t$. We note that the BE holds the inclusion relationships $\tilde{m}(\Phi_{t+k|t}^{j,*}, \hat{u}_{t+k|t}^{j,*}) \subseteq \Phi_{t+k+1|t}^{j,*}, \forall k \in \mathbb{N}_{0:N_t^j-1}$ due to the backward procedure and thus each set calculated by the BE is still a subset of the reachable set to $\mathcal{O}$.

2.3.3 Terminal cost Q^j

Next, we explain how to construct the terminal cost $Q^j(\cdot)$. Given the terminal set $\mathcal{SS}^j$, we define the cost-to-go corresponding to each component $\Phi_{t+k|t}^i$ in $\mathcal{SS}^j$ (for the definition of $\Phi_{t+k|t}^i$, please see subsection 2.3.1) as follows:

$$J_{i,t,k} = \sum_{p=k}^{T_t^i} \max_{x \in \Phi_{t+p|t}^i} \ell(x, u_{t+p|t}^i). \quad (2.27)$$

The terminal cost Q^j for an ellipsoidal set $\bar{\Phi} = \mathcal{E}(\bar{p}, \bar{P})$ in state space $\mathcal{X}$ is defined as the following.

$$Q^j(\bar{\Phi}) = \begin{cases} \min_{(i,t,k) \in F^j(\bar{\Phi})} J_{i,t,k} & \text{if } \exists \Phi = \mathcal{E}(p, P) \in \mathcal{SS}^j, \\ & \bar{\Phi} \subseteq \Phi, \bar{p} = p, \\ +\infty & \text{otherwise,} \end{cases} \quad (2.28)$$

$$F^j(\bar{\Phi}) = \{(i, t, k) : i \in [0, j], t \geq 0 \text{ with } \bar{\Phi} \subseteq \Phi_{t+k|t}^i, \bar{p} = p_{t+k|t}^i; \text{ for } \Phi_{t+k|t}^i \in \mathcal{SS}^j\}. \quad (2.29)$$

In addition, the terminal cost at the initial point $x_0^j = x_s$ is similarly defined as $Q^j(x_s) = \min_{i \in [0, j]} J_{i,0,0}$ with $J_{i,0,0} = \ell(x_s, u_{0|0}^i) + \sum_{p=1}^{T_0^i} \max_{x \in \Phi_{p|0}^i} \ell(x, u_{p|0}^i)$.

2.4 Properties

In this section, we show the theoretical properties of the proposed MPC scheme (2.21). Specifically, the recursive feasibility, input-to-state stability, and performance bound of the closed-loop system are discussed in the following subsections.

2.4.1 Recursive feasibility

We first show the recursive feasibility of the proposed scheme. The illustrative explanation of the recursive feasibility is also shown in Fig. 2.4.

Theorem 1. *Suppose that the trained GP model of the system (2.1) is given and the system (2.1) is controlled by executing the control policy (2.25). Let Assumption 1-5 hold, then the optimization problem (2.21) is feasible for all $t \geq 0$ and iteration $j \geq 1$.*

Proof. We show the recursive feasibility at each iteration $j \geq 1$ by induction. From the definition of $\mathcal{SS}^{j-1}$, we have $\mathcal{SS}^0 \subseteq \mathcal{SS}^{j-1}$. Thus, at time 0 in iteration $j \geq 1$, we have a feasible control sequence of the problem (2.21), $u_{0|0}^0, \dots, u_{N_0^j-1|0}^0$ by following Assumption 5.

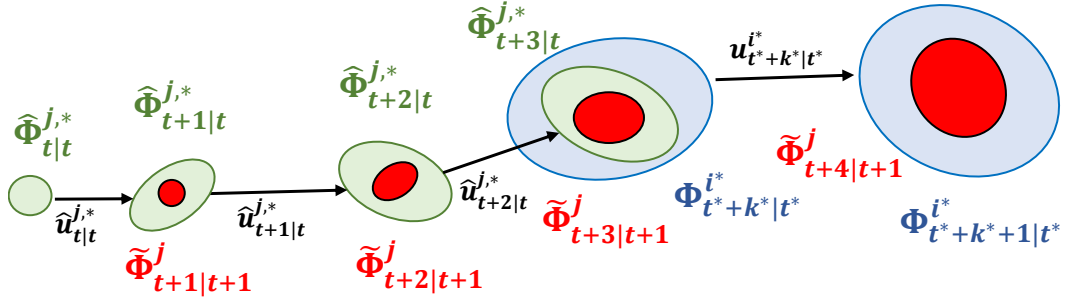


Figure 2.4: The explanation of the recursive feasibility property when $N_t^j = N_t^{j+1} = N$. The blue, green, and red ellipsoids represent the set in $\mathcal{SS}^{j-1}$ ($\Phi_{t^*+k^*|t^*}^{i*}$ is the terminal set selected at time t in iteration j and $\Phi_{t^*+k^*+1|t^*}^{i*}$ is the corresponding one step forwarded set), the optimal sequence of ellipsoids calculated at time t in iteration j , and the sequence of ellipsoids generated by the control inputs $\hat{u}_{t+1|t}^{j,*}, \dots, \hat{u}_{t+N_t^j-1|t}^{j,*}, u_{t^*+k^*|t^*}^{i*}$ at time $t+1$ from the initial ellipsoid $\tilde{\Phi}_{t+1|t+1}^j = \mathcal{B}(\hat{p}_{t|t-1}^{j,*}, |x_t^j - \hat{p}_{t|t-1}^{j,*}|)$, respectively. From the monotonicity property of the propagation of the ellipsoids summarized in Theorem 2 and the definition of the terminal set discussed in subsections 2.3.1 and 2.3.2, the above sequence of control inputs is a feasible solution of the problem (2.21) at time $t+1$ (red ellipsoids are within a blue or green ellipsoid and centers of these ellipsoids are matched).

Next, assume that at time $t > 0$ in iteration $j \geq 1$, the LMPC (2.21) is feasible and let $\hat{u}_{t:t+N_t^j-1|t}^{j,*}$, $\hat{x}_{t+1:t+N_t^j|t}^{j,*}$ and $\hat{\Phi}_{t+1:t+N_t^j|t}^{j,*}$ be the optimal sequences. The terminal constraint (2.21g) enforces $\hat{\Phi}_{t+N_t^j|t}^{j,*} \subset \Phi_{t^*+k^*|t^*}^{i*}$, where $(i^*, t^*, k^*) = \operatorname{argmin}_{(i,t,k) \in F^j(\hat{\Phi}_{t+N_t^j|t}^{j,*})} J_{i,t,k}$.

Then, at time t in iteration j , the control input (2.25) is applied to the system (2.1), and the system state at the next time step is obtained as x_{t+1}^j . Here, we denote the sequence of the confidence regions obtained by applying the control inputs $\hat{u}_{t+1:t+N_{t+1}^j-1|t}^{j,*}$ from the initial ellipsoid at time $t+1$ $\tilde{\Phi}_{t+1|t+1}^j = \mathcal{B}(\hat{p}_{t|t-1}^{j,*}, |x_t^j - \hat{p}_{t|t-1}^{j,*}|) = \mathcal{E}(\tilde{p}_{t+1|t+1}^j, \tilde{P}_{t+1|t+1}^j)$ as $\tilde{\Phi}_{t+1:t+N_{t+1}^j|t+1}^j$. Since the initial ellipsoid has properties $\tilde{\Phi}_{t+1|t+1}^j \subseteq \hat{\Phi}_{t+1|t}^{j,*}$ and $\tilde{p}_{t+1|t+1}^j = \hat{p}_{t+1|t}^{j,*}$, we have $\tilde{\Phi}_{k|t+1}^j \subseteq \hat{\Phi}_{k|t}^{j,*}$ and $\tilde{p}_{k|t+1}^j = \hat{p}_{k|t}^{j,*}, \forall k \in \mathbb{N}_{t+1:t+N_{t+1}^j}$ from the definition and monotonicity property of the confidence region (see subsection 2.1.3 and Lemma 2). Thus, if $N_t^j = N_{t+1}^j + 1$, the control input sequence $\hat{u}_{t+1:t+N_{t+1}^j-1|t}^{j,*}$ satisfies all the constraints in (2.21) and is a feasible solution to the problem (2.21) at time $t+1$. Moreover, when $N_t^j = N_{t+1}^j = N$, since $\tilde{\Phi}_{t+N_{t+1}^j|t+1}^j \subseteq \hat{\Phi}_{t+N_t^j|t}^{j,*} \subseteq \Phi_{t^*+k^*|t^*}^{i*}$ holds and the centers of these ellipsoids are the same, the control input $u_{t^*+k^*|t^*}^{i*}$ brings the confidence region $\tilde{\Phi}_{t+N_{t+1}^j|t+1}^j$ to the inside of $\Phi_{t^*+k^*+1|t^*}^{i*}$ (i.e., $\tilde{m}(\tilde{\Phi}_{t+N_{t+1}^j|t+1}^j, u_{t^*+k^*|t^*}^{i*}) \subseteq \Phi_{t^*+k^*+1|t^*}^{i*}$) and centers of these ellipsoids match from the monotonicity property of the confidence ellipsoids and the definition of sets within $\mathcal{SS}^{j-1}$ discussed in subsections 2.3.1 and 2.3.2. Thus, the sequence of control inputs $\hat{u}_{t+1|t}^{j,*}, \dots, \hat{u}_{t+N_t^j-1|t}^{j,*}, u_{t^*+k^*|t^*}^{i*}$ satisfies all the constraints in (2.21) and is a feasible solution to the

problem (2.21) at time $t + 1$. Then, by induction, we can conclude that the LMPC (2.21) is feasible $\forall j \geq 1$ and $t \geq 0$. $\square$

2.4.2 Input to state stability of the invariant set $\mathcal{O}$

Next, we discuss the stability property of the proposed method. Specifically, ISS is shown for the proposed control scheme. Moreover, we show that the system states converge to the set $\mathcal{O}$ with a specific choice of the terminal set. For the input-to-state stability and analysis of the performance-bound discussed in the next subsection, we require the following assumption on the terminal set $\mathcal{SS}^j$.

Assumption 6. *All the ellipsoidal sets within $\mathcal{SS}^j$, $\forall j$ have different centers.*

Note that Assumption 6 is required only for the ISS and performance cost bound (Theorem 2 and 3), and the other properties including the recursive feasibility (Theorem 1) and the convergence properties (Remark 1 and 2) hold without Assumption 6. As a tool to check the ISS property of the closed loop system with respect to the control invariant set $\mathcal{O}$, we first introduce the dissipative-form ISS-Lyapunov function.

Definition 2. *A dissipative-form ISS-Lyapunov function for the closed-loop system and the invariant set $\mathcal{O}$ is a function $V : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ such that there exist $\alpha_1, \alpha_2, \alpha \in \mathcal{K}_{\infty}$, and $\sigma \in \mathcal{K}$ so that for all $x_t^j \in \mathbb{R}^n$ and the one-step prediction error of the certainty equivalent prediction e_t^j ,*

$$\alpha_1(|x_t^j|_{\mathcal{O}}) \leq V(x_t^j) \leq \alpha_2(|x_t^j|_{\mathcal{O}}), \quad (2.30)$$

$$V(f(x_t^j, u_t^j)) - V(x_t^j) \leq -\alpha(|x_t^j|_{\mathcal{O}}) + \sigma(\|e_t^j\|). \quad (2.31)$$

The following proposition gives the relationship between the above ISS-Lyapunov function and ISS of $\mathcal{O}$.

Proposition 1. *The following statements are equivalent:*

- *The close loop system is ISS with respect to the robust invariant set $\mathcal{O}$.*
- *There exists a dissipative-form ISS-Lyapunov function.*

Proof. The proof follows from Theorem 2.3 in [128]. $\square$

We note here that the continuity of the function V is not required in [128]. Then, the ISS property holds for the proposed control scheme as summarized in the following theorem.

Theorem 2. Suppose the GP model is given, and consider the closed-loop system (2.1) and (2.25). Let Assumption 1-6 hold. Then, the closed-loop system is ISS for the set $\mathcal{O}$.

Proof. In the following, we first consider the case when $N_t^j = N_{t+1}^j = N$, and the other cases are subsequently discussed. The LMPC cost at time t can be written as

$$\begin{aligned} J_{t \rightarrow t+N}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*}) &= \sum_{k=t}^{t+N-1} \ell(\hat{x}_{k|t}^{j,*}, \hat{u}_{k|t}^{j,*}) + Q^{j-1}(\hat{\Phi}_{t+N|t}^{j,*}) \\ &= \ell(\hat{x}_{t|t}^{j,*}, \hat{u}_{t|t}^{j,*}) + p(\hat{x}_{t+1|t}^{j,*}) + Q^{j-1}(\hat{\Phi}_{t+N|t}^{j,*}), \end{aligned} \quad (2.32)$$

where $p(\hat{x}_{t+1|t}^{j,*})$ represents the total cost from time $t+1$ to $t+N-1$ when the control inputs $[\hat{u}_{t+1|t}^{j,*}, \dots, \hat{u}_{t+N-1|t}^{j,*}]$ are applied to the certainty equivalent prediction model. After the control input (2.25) is applied to the system (2.1), the next state is obtained as x_{t+1}^j . Then, from the discussion in subsection 2.4.1, the sequence of the control inputs $[\hat{u}_{t+1|t}^{j,*}, \dots, \hat{u}_{t+N-1|t}^{j,*}, u_{t^*+k^*|t^*}^{i^*}]$, where $(i^*, t^*, k^*) = \text{argmin}_{(i,t,k) \in F^j(\hat{\Phi}_{t+N|t}^{j,*})} J_{i,t,k}$, is a feasible solution to the problem (2.21) at time $t+1$ and we denote the associated mean predictions and confidence regions of system states as $[\tilde{x}_{t+1|t+1}^j, \dots, \tilde{x}_{t+N+1|t+1}^j]$ with $\tilde{x}_{t+1|t+1}^j = x_{t+1}^j$ and $[\tilde{\Phi}_{t+1|t+1}^j, \dots, \tilde{\Phi}_{t+N+1|t+1}^j]$ with $\tilde{\Phi}_{t+1|t+1}^j = \mathcal{B}(\hat{p}_{t|t-1}^{j,*}, |x_t^j - \hat{p}_{t|t-1}^{j,*}|)$. Then, the cost of the LMPC at time $t+1$ satisfies the following inequality

$$\begin{aligned} J_{t+1 \rightarrow t+1+N}^{\text{LMPC},j}(x_{t+1}^j, \hat{p}_{t+1|t}^{j,*}) &\leq \sum_{k=t+1}^{t+N-1} \ell(\tilde{x}_{k|t+1}^j, \hat{u}_{k|t}^{j,*}) + \ell(\tilde{x}_{t+N|t+1}^j, \hat{u}_{t^*+k^*|t^*}^{i^*}) + Q^{j-1}(\tilde{\Phi}_{t+N+1|t+1}^j) \\ &\leq \sum_{k=t+1}^{t+N-1} \ell(\tilde{x}_{k|t+1}^j, \hat{u}_{k|t}^{j,*}) + Q^{j-1}(\tilde{\Phi}_{t+N|t+1}^j) = p(x_{t+1}^j) + Q^{j-1}(\tilde{\Phi}_{t+N|t+1}^j), \end{aligned} \quad (2.33)$$

where the last inequality follows from

$$\begin{aligned} Q^{j-1}(\tilde{\Phi}_{t+N|t+1}^j) &= \sum_{p=k_2^*}^{T_{t_2^*}^{j*}} \max_{x \in \Phi_{t_2^*+p|t_2^*}^{i_2^*}} \ell(x, u_{t_2^*+p|t_2^*}^{i_2^*}) = \sum_{p=k^*}^{T_{t^*}^{i^*}} \max_{x \in \Phi_{t^*+p|t^*}^{i^*}} \ell(x, u_{t^*+p|t^*}^{i^*}) \\ &\geq \ell(\tilde{x}_{t+N|t+1}^j, u_{t^*+k^*|t^*}^{i^*}) + \sum_{p=k^*+1}^{T_{t^*}^{i^*}} \max_{x \in \Phi_{t^*+p|t^*}^{i^*}} \ell(x, u_{t^*+p|t^*}^{i^*}) \\ &= \ell(\tilde{x}_{t+N|t+1}^j, u_{t^*+k^*|t^*}^{i^*}) + Q^{j-1}(\tilde{\Phi}_{t+N+1|t+1}^j), \end{aligned} \quad (2.34)$$

where $(i_2^*, t_2^*, k_2^*) = \text{argmin}_{(i,t,k) \in F^j(\hat{\Phi}_{t+N|t+1}^j)} J_{i,t,k}$, the second equality follows from the definition of the terminal cost discussed in subsection 2.3.1, Assumption 6, and the relationships of the ellipsoids $\tilde{\Phi}_{t+N|t+1}^j \subseteq \Phi_{t^*+k^*|t^*}^{i^*}$ and $\tilde{p}_{t+N|t+1}^j = p_{t^*+k^*|t^*}^{i^*}$. The first inequality follows from $\tilde{x}_{t+N|t+1}^j \in \Phi_{t^*+k^*|t^*}^{i^*}$, and the last equality follows from the property discussed in Lemma 2, Assumption 6, and $\tilde{\Phi}_{t+N+1|t+1}^j \subseteq$

$\Phi_{t^*+k^*+1|t^*}^{i^*}$ and centers of these ellipsoids match. Then, from (2.32) and (2.33), we have

$$\begin{aligned}
& J_{t+1 \rightarrow t+1+N}^{\text{LMPC},j}(f(x_t^j, u_t^j), \hat{p}_{t+1|t}^{j,*}) - J_{t \rightarrow t+N}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*}) \\
& \leq p(\hat{x}_{t+1|t}^{j,*} + e_t^j) - p(\hat{x}_{t+1|t}^{j,*}) - \ell(\hat{x}_{t|t}^{j,*}, u_{t|t}^{j,*}) \\
& \leq -\ell(\hat{x}_{t|t}^{j,*}, u_{t|t}^{j,*}) + L\|e_t^j\| \leq -\ell(\hat{x}_{t|t}^{j,*}, 0) + L\|e_t^j\| \\
& \leq -\alpha_x^j(x_t^j) + L\|e_t^j\|,
\end{aligned} \tag{2.35}$$

where we used $Q^{j-1}(\hat{\Phi}_{t+N|t}^{j,*}) = Q^{j-1}(\tilde{\Phi}_{t+N|t+1}^j)$ which holds from Assumption 6, Lemma 2, and $\tilde{\Phi}_{t+N|t+1}^j \subseteq \hat{\Phi}_{t+N|t}^{j,*}$ and $\tilde{p}_{t+N|t+1}^j \subseteq \hat{p}_{t+N|t}^{j,*}$. L is the Lipschitz constant of the function p and e_t^j is the prediction error of $\hat{f}$ (i.e., $e_t^j = x_{t+1}^j - \hat{x}_{t+1|t}^{j,*}$). The last inequality follows from Assumption 4. Since the condition (2.30) is satisfied irrespective of the values of $\hat{p}_{t|t-1}^{j,*}$ and $\hat{p}_{t+1|t}^{j,*}$, we can confirm that the condition (2.30) is satisfied if $N_t^j = N_{t+1}^j = N$ by defining $V(x_t^j) = J_{t \rightarrow t+N}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*})$. Next, we consider the case $N_t = N_{t+1} + 1$. In this case, the inequality (2.35) is obviously satisfied since $Q^j(\tilde{\Phi}_{t+N_{t+1}^j|t+1}^j) = 0$ holds from the fact $\tilde{\Phi}_{t+N_{t+1}^j|t+1}^j \subseteq O$. $\square$

Remark 1. Although the above ISS property does not imply that the system states always converge to the set O with the control inputs calculated by the proposed MPC strategy (2.25), there always exists a sequence of control inputs that steer the system states to the set O , which first goes to the currently chosen terminal set $\Phi_{t^*+k^*|t^*}^{i^*}$ (i.e., $\hat{u}_{t:t+N_t^j-1|t}^{j,*}$) and then goes to O (i.e., $u_{t^*+k^*:t^*+T_{t^*}^{i^*}|t^*}$).

Remark 2. In addition to the ISS of the set O , we can further show that the system states converge to the set O with the control input (2.25) by restricting the candidate terminal set within $\mathcal{SS}^j$ when the optimization (2.21) is solved so that only the sets with terminal costs lower than that of the terminal set selected in the previous time step are considered. With this scheme, we can ensure that the terminal cost is monotonically decreasing with respect to the time step, and thus the terminal cost eventually be 0 (i.e., O is selected as the terminal set in finite time). Then, the convergence to the set O is guaranteed from the definition of the MPC problem (2.21).

2.4.3 Performance bound

Lastly, we discuss the bound on the performance cost when the proposed LMPC strategy (2.25) is executed on the system (2.1) at j -th iteration.

Theorem 3. Let Assumption 1-6 hold. Then, the upper bound on the iteration cost of the close-loop system (2.1) and (2.25) is obtained as the following:

$$J_{0 \rightarrow T^j}^j(x_0^j) := \sum_{t=0}^{T^j} \ell(x_t^j, u_t^j) \leq Q^{j-1}(x_0^j) + L \sum_{t=0}^{T^j} \|e_t^j\|,$$

where $e_t^j = x_t^j - \hat{x}_{t|t-1}^{j,*}$ and L is defined in the proof of Theorem 2.

Proof. From the definition of Q^{j-1} discussed in subsection 2.3.3, we have

$$\begin{aligned} Q^{j-1}(x_0^j) &= \ell(x_0^j, u_{0|0}^{i*}) + Q^{j-1}(\Phi_{1|0}^{i*}) = \ell(x_0^j, u_{0|0}^{i*}) + \max_{x \in \Phi_{1|0}^{i*}} \ell(x, u_{1|0}^{i*}) + Q^{j-1}(\Phi_{2|0}^{i*}) \\ &\geq \ell(x_0^j, u_{0|0}^{i*}) + \ell(\tilde{x}_1^{i*}, u_{1|0}^{i*}) + Q^{j-1}(\Phi_{2|0}^{i*}) \\ &\geq \sum_{k=0}^{N_0^j-1} \ell(\tilde{x}_k^{i*}, u_{k|0}^{i*}) + Q^{j-1}(\Phi_{N_0^j|0}^{i*}) \geq J_{0 \rightarrow N_0^j}^{\text{LMPC},j}(x_0^j, x_0^j), \end{aligned} \quad (2.36)$$

where $i^* = \arg\min_{i \in [0,j]} J_{0 \rightarrow \infty}^i(x_s)$ and $\tilde{x}_k^{i*}$ is the state that is obtained by applying control inputs $u_0^{i*}, \dots, u_{k-1}^{i*}$ to the dynamics (2.1). The last inequality follows from the fact that the sequence of control inputs $u_{0:N_0^j-1}^{i*}$ is a feasible solution to $J_{0 \rightarrow N_0^j}^{\text{LMPC},j}(x_0^j)$. Moreover, from (2.35), we have

$$J_{t+1 \rightarrow t+1+N_{t+1}^j}^{\text{LMPC},j}(x_{t+1}^j, \hat{p}_{t+1|t}^{j,*}) - J_{t \rightarrow t+N_t^j}^{\text{LMPC},j}(x_t^j, \hat{p}_{t|t-1}^{j,*}) \leq -\ell(\hat{x}_{t|t}^{j,*}, \hat{u}_t^j) + L\|e_t^j\|. \quad (2.37)$$

By iteratively applying (2.37), we have

$$\begin{aligned} J_{0 \rightarrow N_0^j}^{\text{LMPC},j}(x_0^j, x_0^j) &\geq \sum_{t=0}^{T^j} \ell(x_t^j, u_t^j) - L \sum_{t=0}^{T^j} \|e_t^j\| + J_{T^j+1 \rightarrow T^j+1+N}^{\text{LMPC},j}(x_{T^j+1}^j, \hat{p}_{T^j+1|T^j}^{j,*}) \\ &\geq J_{0 \rightarrow T^j}^j(\hat{x}_0^j) - L \sum_{t=0}^{T^j} \|e_t^j\|, \end{aligned} \quad (2.38)$$

where the last inequality follows from the Lipschitz continuity of the function ℓ . From (2.36) and (2.38), we have

$$Q^{j-1}(x_0^j) \geq J_{0 \rightarrow N_0^j}^{\text{LMPC},j}(x_0^j, x_0^j) \geq J_{0 \rightarrow T^j}^j(x_0^j) - L \sum_{t=0}^{T^j} \|e_t^j\|. \quad (2.39)$$

This concludes the proof. $\square$

Regarding the above performance bound, we have the following: (i) The part $Q^{j-1}(x_0^j)$ is improved when the iteration proceeds from the definition of $Q^j(\cdot)$ discussed in subsection 2.3.1. (ii) If the error associated with the GP prediction becomes small, the part $L \sum_{t=0}^{T^j} \|e_t^j\|$ becomes small, and thus the upper bound of the performance cost is also reduced.

2.5 Practical Considerations

2.5.1 Control scheme with GP update

If the GP models are updated during the execution, the properties discussed in Section 2.4 do not hold anymore with the terminal components $\mathcal{SS}^j$ and Q^j defined using the past GP models since a control

input collected in a past iteration $u_{t+k|t}^i$ does not necessarily bring the confidence ellipsoids of a new GP model from $\Phi_{t+k|t}^i$ to the inside of $\Phi_{t+k+1|t}^i$. Thus, we need to recalculate the terminal components with the updated GP model. Similar to the discussion of the BE in subsection 2.3.2, the recalculation of the terminal sets can be done by solving the following problem backward in time (from time $t + N_t^j$ to t) with $\Phi_{t+N_t^j|t}^{j,\text{new}} = \Phi_{t^*+k^*|t^*}^{i^*,\text{new}}$, where (t^*, k^*, t^*) is the triple of integers with $\Phi_{t+N_t^j|t}^j = \Phi_{t^*+k^*|t^*}^{i^*}$ (see also subsection 2.3.2):

$$\Phi_{t+k|t}^{j,\text{new}} = \arg\max_{\Phi} \text{Vol}(\Phi) = \arg\max_{\Phi} \text{tr}(L_P), \quad (2.40a)$$

$$\text{s.t. } \Phi = \mathcal{E}(p_{t+k|t}^j, P), \quad P = L_P L_P^\top, \Phi \subseteq \mathcal{X}, \quad (2.40b)$$

$$\mathcal{E}(p', P') := \tilde{m}(\Phi, \hat{u}_{t+k|t}^j), \quad (2.40c)$$

$$P_{t+k+1|t}^{j,\text{new}} - P' \succeq 0, \quad p_{t+k+1|t}^{j,\text{new}} = p', \quad (2.40d)$$

where $\Phi_{t+k|t}^{j,\text{new}}$ is the new ellipsoid for the updated GP model. This procedure is executed from $j = 0$. With this recalculation, the desired inclusion relations $\tilde{m}(\Phi_{t+k|t}^{j,\text{new}}, \hat{u}_{t+k|t}^{j,*}) \subseteq \Phi_{t+k+1|t}^{j,\text{new}}$ hold.

2.5.2 Construction of the initial sets

The initial ellipsoids in Assumption 5 can be obtained with the BE procedure discussed in subsection 2.3.2, given control input sequence. If the growth of the confidence ellipsoids along with the time steps is fast and the required number of time steps to solve the original control problem (2.21) is large, it may not be easy to construct the initial sets that meet inclusion relation in Assumption 5. Then, the designer may want to instead use the initial sets $\Phi_{1|0}^0, \Phi_{2|0}^0, \dots, \Phi_{T^0|0}^0 \subseteq \mathcal{X}$ that do not necessarily satisfy the inclusion relations $\Phi_{t+1|0}^0 \supseteq \tilde{m}(\Phi_{t|0}^0, u_{t|0}^0)$. In this case, the theoretical results discussed in Section 2.4 do not hold anymore. Instead, the following properties hold for all time $t > 0$ and iteration $j > 1$: (i) There always exists a sequence of the control inputs that bring the confidence ellipsoidal set to the one of the initial safe set $\Phi_{t|0}^0, \exists t \in \mathbb{N}_{0:T^0}$ (ii) If the terminal set selected in the previous time step is not a set within the initial sets, the feasibility of the problem (2.21) at that time is guaranteed. Thus, if the initial sets $\Phi_{t|0}^0, \exists t \in \mathbb{N}_{0:T^0}$ are somehow guaranteed to be safe (e.g., there exist sequences of control inputs that safely bring the state within the initial sets to the initial state or the target set O), our method can still provide safety while allowing to iteratively improve the terminal sets of the MPC problem by effectively utilizing the data obtained in the previous iterations. In the case study in Section 2.6, we have empirically confirmed that the proposed control scheme with the above case of initial sets can attain monotonically decreasing performance cost without experiencing infeasibility during the execution.

2.6 Case Study

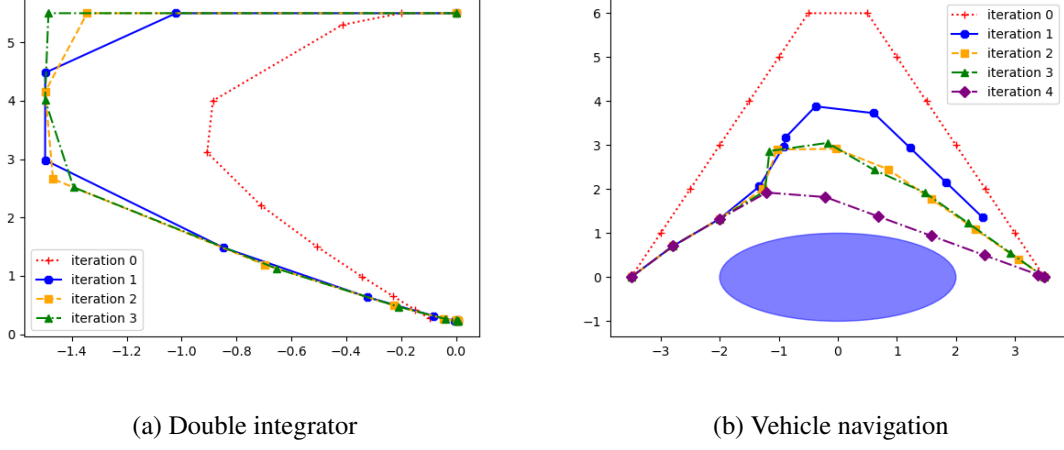


Figure 2.5: The actual trajectories obtained by executing the proposed method. We can see that the performance of the closed-loop system is improved along with the iterations.

In this section, we show the efficacy of the proposed method through the examples of a double integrator system and vehicle navigation problem. Specifically, we investigate whether the proposed iterative MPC scheme can improve the performance cost along with the number of iterations. All the experiments are conducted on Python, and the IPOPT solver in CasADi [129] is used to solve all of the optimization problems in the experiments. For training the GP model, we used the GPy library [130], and the GP uncertainty propagation model is implemented based on [131]. Since the uncertainty ellipsoids are overly conservative if we directly use the actual values of the constant β and Lipschitz constants L_σ and $L_{\nabla\mu_g}$, we set β and each element of L_σ and $L_{\nabla\mu_g}$ to 1, 0.001 and 0.01, respectively for both cases. Such reduction is also employed in previous studies such as [55, 131]. We also set the maximum horizon length N and the execution time T^j , $\forall j \geq 1$ to 4 and 10, respectively, for both cases. Although the optimization problem (2.21) can be formulated as a non-linear mixed integer program similar to [49], we instead implement the problem (2.21) by separately solving all of the optimization problems corresponding to each candidate terminal set in $\mathcal{SS}^{j-1}$ and using the solution with the smallest cost. Although this procedure requires solving many optimization problems at each time instance, we note that the optimizations for the candidate sets can be solved in parallel.

Table 2.1: Performance cost of the proposed control scheme at each iteration (double integrator)

$j = 0$	$j = 1$	$j = 2$	$i = 3$	$i = 4$
128.267	102.754	98.8373	97.5147	97.5147

2.6.1 Double integrator

As an initial experiment, we consider the control problem of the double integrator system with the dynamics:

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t. \quad (2.41)$$

The dynamics are assumed to be fully unknown and learned with GP. The GP model is trained with 160 pairs of a control input and system state randomly collected from the intervals $[-1, 6] \times [-1.5, 1.5]$ and $[-1.5, 1.5]$, respectively, and the corresponding outputs of the dynamics above. The cost function is defined as $\ell(x, u) = x^\top x + 2u^2$ and the set $\mathcal{O}$ is the disk with a radius of 1.5 centered at the origin. Although the value of the cost function should be zero for all pairs of $x \in \mathcal{O}$ and $u \in \mathcal{U}$ according to Assumption 3, we use the above cost for simple implementation in this experiment. The constraint sets for control input and the second element of system states are given by $\mathcal{U} = \{u \mid |u| < 1.5\}$ and $\mathcal{X} = \{x \mid |x_{t,2}| < 1.5\}$, $\forall t > 0$, respectively. The initial ellipsoids mentioned in Assumption 5 are constructed by first generating the control inputs that correspond to the state transition of the red points in Fig. 2.5 (a) and then executing the computation similar to the BE from the set $\mathcal{O}$. Then, we execute Algorithm 1. More specifically, at each time in each iteration, the optimization problem (2.21) is solved and the control input (2.25) is applied to the actual system (2.1), and then the next state is observed. After each iteration, the terminal set and cost are updated with the collected ellipsoids and control inputs with the procedures discussed in subsections 2.3.1, 2.3.2, and 2.3.3. The results are shown in Table 2.1 and Fig. 2.5. In Table 2.1, the performance cost of the proposed control scheme at each iteration is summarized. As we can see from the table, the iteration cost is monotonically decreased until it reaches the steady state value at iteration 4. In Fig. 2.5 (a), the state trajectories obtained by applying the control inputs (2.25) to the actual system (4.1) are shown. We can also confirm the improvement of the control performance from the figure.

Table 2.2: Performance cost of the proposed control scheme at each iteration (vehicle navigation)

$j = 0$	$j = 1$	$j = 2$	$i = 3$	$i = 4$
423.5	279.935	226.057	220.627	179.76

2.6.2 Vehicle navigation

In the second experiment, we consider vehicle navigation in a 2D space. The dynamics of the vehicle are given by

$$x_{t+1} = x_t + \begin{bmatrix} \cos \theta_t & 0 \\ \sin \theta_t & 0 \\ 0 & 1 \end{bmatrix} u_t, \quad (2.42)$$

where the state $x_t \in \mathbb{R}^3 = [q_{x,t}, q_{y,t}, \theta_t]^\top$ consists of the position in an x - y coordinate ($q_{x,t}, q_{y,t}$) and heading angle θ_t . The control input $u_t = [v_t, \omega_t]^\top \in \mathbb{R}^2$ includes the speed v_t and angular speed ω_t . The GP model is trained with 160 pairs of control inputs and system states randomly collected from the intervals $[-4, 4] \times [-0.5, 6.5] \times [-\pi/2, \pi/2]$ and $[-1.3, 1.3] \times [-1.3, -1.3]$, respectively, and the corresponding outputs of the dynamics above. The cost function is defined as $\ell(x, u) = \|x - x_{\text{target}}\|^2$ with the target state $x_{\text{target}} = [3.5, 0, 0]^\top$. The initial state is set to $x_s = [-3.5, 0, \pi/4]^\top$ and the goal set $\mathcal{O}$ is defined by the disk with a radius of 0.75 centered at x_{target} . We define the initial terminal sets by the balls with a radius of 0.75 centered around the pairs of x - y position represented by a red cross in Fig. 2.5 (b) and a heading angle ($\pi/3$ for the first 5 red crosses, 0 for the next 2 red crosses, and $-\pi/3$ for the rest). This is the case discussed in subsection 2.5.2. The obstacle that is required to be avoided for all the execution time is the 2-D ellipsoid shown in Fig. 2.5. Since it is hard to obtain an analytical expression of the condition that an uncertainty ellipsoid is outside of the obstacle, we instead calculate the maximum and minimum x, y values within the uncertainty ellipsoid and impose the constraints that all of the combinations of these x, y values are outside of the obstacle. Moreover, we also impose the constraints for the velocity and angular velocity as $|v_t| < 1.2$ and $|\omega_t| < 1.2$, respectively. The results are shown in Fig. 2.5 (b), and Table 2.2. From these, we can see that the cost is improved along with iteration.

2.7 Summary

In this study, we considered an iterative MPC scheme with the learned dynamics model. Specifically, we proposed how to construct and improve the terminal components (terminal cost and set) of the MPC problem by effectively utilizing the data collected from the past iterations. To overcome the practical

challenges associated with the proposed control scheme, we additionally introduced some techniques such as backward enlargement (BE) of reachable sets. Then, we showed that the proposed MPC scheme meets recursive feasibility and the ISS properties under certain assumptions. With further restrictions on the proposed scheme, the convergence to the set $\mathcal{O}$ with the proposed MPC scheme can also be guaranteed. Moreover, we analyzed the performance bound of the resulting closed-loop system. The results from the case study show that the proposed method can improve the performance cost along with iterations for some control problems. In future research, we will explore approaches to reduce the conservativeness of the confidence ellipsoids, aiming to enhance the applicability of the proposed MPC scheme.

Chapter 3

Data-Efficient Safe Learning and Control with On-Board Sensors: Bayesian Meta-Learning and Barrier Function Based Approach

We consider the safe navigation of robots in unknown environments using data from sensory devices. The control barrier function (CBF) is one of the promising approaches to enforcing safety in robot navigation and the recent progress on learning-based approaches realizes online synthesis of CBF-based safe controllers with sensor measurements. However, the learned CBF candidates in these works cannot be generalized to different environments and the re-synthesis is necessary when changes in the environment occur. With this observation, this study attempts to develop a method that can effectively use past data in different environments to quickly learn the CBF candidate in the current setting by leveraging the currently developed Bayesian meta-learning framework. Our method realizes data-efficient online learning as shown in the simulation and provides safety guarantees on the resulting controller.

This chapter is organized based on “Data-efficient safe learning and control with on-board sensors: Bayesian meta-learning and barrier function based approach ” [132] which appeared in Advanced Robotics.

Notations: Continuous functions $\alpha_1 : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ and $\alpha_2 : \mathbb{R} \rightarrow \mathbb{R}$ are class $\mathcal{K}$ and extended class $\mathcal{K}_{\infty}$ functions, respectively, if they are strictly increasing with $\alpha_1(0) = 0$ and $\alpha_2(0) = 0$. $\chi_d^2(p)$ is the p -th quantile of the χ^2 distribution with d degrees of freedom. The maximum and minimum

eigenvalues for any positive definite matrix A are defined as $\bar{\lambda}(A)$ and $\underline{\lambda}(A)$, respectively. For a vector field f and a function F , $L_f F(x) := \frac{\partial F(x)}{\partial x} f(x)$ is the Lie derivative of the function F in the direction of f .

3.1 Problem Statement

Throughout this chapter, we consider the following control affine system¹:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \quad (3.1)$$

where $x(t) \in \mathcal{D} \subset \mathbb{R}^n$ and $u(t) \in \mathcal{U} := \{u \in \mathbb{R}^m \mid Au \leq b\}$ with $A \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ are the admissible system states and control inputs at time t , respectively. The functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are continuously differentiable functions representing dynamics of a robotic system (e.g., ground vehicles or drones) equipped with sensors such as LiDAR. The state space is assumed to contain *initially unknown* N_{obs} obstacles and the *outside* of each obstacle is defined by the following:

$$C_{\xi_i} = \{x \in \mathcal{D} \subset \mathbb{R}^n \mid h(x; \xi_i) \geq 0\}, \quad i \in \mathbb{N}_{1:N_{\text{obs}}}, \quad (3.2)$$

where $h(\cdot; \xi) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuously differentiable function with *unobserved* latent variable ξ that encodes the geometric information about an obstacle, e.g., the position and shape of an obstacle. We assume realizations of the variable, $\xi_i, i \in \mathbb{N}_{1:N_{\text{obs}}}$ are determined by samples from a probabilistic distribution $p(\xi)$ and they are fixed during the control execution. For simplicity, the obstacles are chosen not to overlap with each other.

Our goal is to synthesize a controller that achieves a given task (e.g., goal-reaching task) without deviating from the safe region $C = \bigcup_{i=1}^{N_{\text{obs}}} C_{\xi_i}$ given initial state $x(0) \in C$. Since the safe/unsafe regions are initially unknown, the robotic system needs to identify them based on the sensor measurements. A naive approach is to use supervised learning based on measurement data obtained under a pre-determined environment. However, in this case, the trained model cannot be generalized to different environments, and even slight changes in the environment necessitate a complete re-fitting of the model, which is not only inefficient but also demands a substantial amount of data. Thus, we consider a way to effectively use past data in the different environments sampled from $p(\xi)$ to quickly learn safe/unsafe regions in a new environment and synthesize the safe controller with few amount of online data.

¹ Although our proposed method is applicable to broader classes of non-linear systems, we focus on control affine systems because this approach simplifies the control problem into a QP, which can be solved efficiently. Note that this limitation is typical in the literature on CBF (see, e.g., [26]).

3.2 CLF-CBF-QP

In this section, we first introduce the Zeroing Control Barrier Function (ZCBF) as a tool to enforce safety on the system (3.1). The following discussion is a summary of the works [26, 30]. In ZCBF, the safety is defined as the forward-invariance of a safe set C which is defined by the zero super level set of a function $h : \mathbb{R}^n \rightarrow \mathbb{R}$, i.e., the system (3.1) is considered to be safe if there exists a Lipschitz continuous control $u(t)$ that results in $x(t) \in C := \{x \mid h(x) \geq 0\}$ for all $t \geq 0$ when $x(0) \in C$. Given an extended class $\mathcal{K}_\infty$ function $\alpha_C : \mathbb{R} \rightarrow \mathbb{R}$, the ZCBF is defined as follows.

Definition 3. Let $C \subset \mathcal{D} \subset \mathbb{R}^n$ be the zero super level set of a continuously differentiable function $h : \mathbb{R}^n \rightarrow \mathbb{R}$. Then, the function h is a zeroing control barrier function (ZCBF) for (3.1) on C if there exists an extended class $\mathcal{K}_\infty$ function α_C such that, for all $x \in \mathcal{D}$, the following inequality holds.

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x)u + \alpha_C(h(x))\} \geq 0, \quad (3.3)$$

where $L_f h(x)$ and $L_g h(x)$ are the Lie derivatives of the function h in the direction of the vector fields f and g , which are defined in notations in the beginning of this chapter. We additionally define the set of control inputs that satisfy the CBF condition as follows:

$$U_C(x) = \{u \in \mathcal{U} \mid L_f h(x) + L_g h(x)u + \alpha_C(h(x)) \geq 0\}.$$

If the function h is a valid CBF for (3.1) on C , the safety of the system (3.1) is guaranteed as the following theorem.

Theorem 4 ([30]). Let $C \subset \mathcal{D} \subset \mathbb{R}^n$ be a zero super level set of a continuously differentiable function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ and the function h is ZCBF for (3.1). Then, any Lipschitz continuous policy that selects control inputs from $U_C(x)$ renders the safe set C forward invariant for the system (3.1).

In this study, the control objectives, except for the safety requirements, are encoded through the control Lyapunov function (CLF) which is given in advance. With a class $\mathcal{K}$ function α_V , the definition of CLF is formally given as follows [26].

Definition 4. A continuously differentiable function $V : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ is a CLF if $V(x_e) = 0$ and $V(x) > 0$, $x \neq x_e$ with equilibrium point $x_e \in \mathcal{D}$, V is radially unbounded, and there exists a class $\mathcal{K}$ function $\alpha_V : \mathbb{R} \rightarrow \mathbb{R}$ such that, for all $x \in \mathcal{D} \subset \mathbb{R}^n$, the following inequality holds.

$$\inf_{u \in \mathcal{U}} \{L_f V(x) + L_g V(x)u + \alpha_V(V(x))\} \leq 0. \quad (3.4)$$

Given a CLF V , we consider the set of all control inputs that satisfy (3.4) for a state $x \in \mathcal{D}$ as

$$U_V(x) = \{u \in \mathcal{U} \mid L_f V(x) + L_g V(x)u + \alpha_V(V(x)) \leq 0\}.$$

If a function V is CLF, the stability of the equilibrium point x_e is guaranteed as the following theorem.

Theorem 5 ([25]). *Let a continuously differentiable function $V : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ be CLF for (3.1). Then, any Lipschitz continuous policy that selects control inputs from $U_V(x)$ renders the equilibrium point x_e of the system (3.1) asymptotically stable.*

Note that if the class $\mathcal{K}$ function α_V is linear, i.e., $\alpha_V(V(x)) := \gamma V(x)$ with $\gamma > 0$, the condition (3.4) exhibits exponential stability [30]. Having CBF and CLF defined above, we can synthesize a controller satisfying CBF and CLF conditions (3.3) and (3.4) with quadratic programming (QP) as follows:

$$\min_{(u, \epsilon) \in \mathbb{R}^{m+1}} \frac{1}{2} u^\top H u + \lambda \epsilon^2, \quad (3.5a)$$

$$\text{s.t. } L_f V(x) + L_g V(x)u + \alpha_V(V(x)) \leq \epsilon, \quad (3.5b)$$

$$L_f h(x) + L_g h(x)u + \alpha_C(h(x)) \geq 0, \quad (3.5c)$$

$$A u \leq b, \quad (3.5d)$$

where $H \in \mathbb{R}^{m \times m}$ is a positive definite matrix and $\lambda (> 0)$ is a tunable parameter for relaxing the CLF condition (3.5b). The relaxation of the CLF condition ensures the solvability of the QP. If the safety is defined by multiple CBFs, we can handle them by adding the corresponding CBF constraints to the optimization problem. The CLF-CBF-QP controller (3.5) exhibits local Lipschitz continuity as shown in [30] and thus the local existence and uniqueness of solutions to the resulting closed-loop system are guaranteed.

3.3 Bayesian Meta-Learning of CBF Candidate

In this section, we consider learning the safe region (3.2) based on the measurement data from sensor devices. To this end, we employ a technique based on the Gaussian process implicit surfaces (GPIS) [90–92] which is developed for non-parametric probabilistic reconstruction of object surfaces. Note that we consider the case of $N_{\text{obs}} = 1$ for a while, and the case of multiple obstacles is discussed later in the next section. In this study, the implicit surfaces (IS) are defined by the distance from the surface of objects to a certain point z in 2D or 3D space, and the inside and outside of the object surfaces are interpreted through the sign of the function as follows.

$$h_{\text{IS}}(z; \xi) = \begin{cases} d_\xi(z) & \text{if } z \text{ is outside the obstacle} \\ -d_\xi(z) & \text{otherwise} \end{cases} \quad (3.6)$$

where $z \in \mathbb{R}^b$, $b \in \{2, 3\}$ is a position in a 2D or 3D space and $d_\xi : \mathbb{R}^b \rightarrow \mathbb{R}_{\geq 0}$ is the function that returns the minimum Euclidean distance between z and surface of the obstacle corresponding

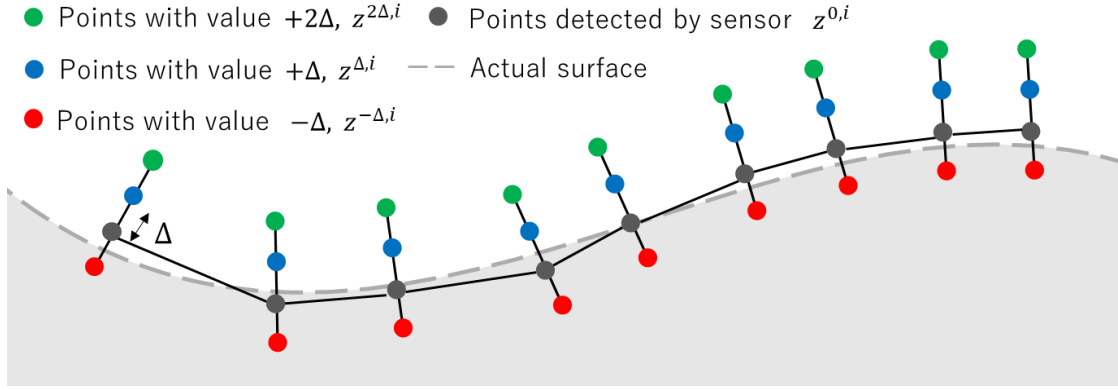


Figure 3.1: The construction of data points from noisy sensor measurements: The gray dots are the points on the surface detected by the sensors. The green, blue, and red dots represent the points that have distance 2Δ , Δ , $-\Delta$ from the surface (negative value means inside of the obstacle) which are calculated by using surface normal approximation explained in subsection 3.3.1 (we show the case of $n_- = 1$ and $n_+ = 2$).

to $\xi \sim p(\xi)$. Though this study focuses on a 2D environment, our method can easily be extended to a 3D case. In GPIS, the function h_{IS} is estimated with the GP, which enables us to deal with uncertainty arising from measurement noise and scarcity of data. The GP is estimated with a dataset $\{(z^i, h_{\text{IS}}(z^i, \xi) + \varsigma_i)\}_{i=1}^{n_{\text{data}}}$, where ς_i is bounded σ -subgaussian noise and n_{data} is the number of data.

Remark 3. *Although this study considers a learning method for separated obstacles, our proposed learning method introduced in this section can potentially be extended to scene-level learning. Since this extension is ideal for many practical applications, we will consider it in future work.*

3.3.1 Dataset construction

In this subsection, we explain how to construct a dataset to learn implicit surfaces (3.6) based on the measurement data from sensors such as LiDAR. A pictorial image for obtaining training data from noisy sensor measurements is shown in Fig. 3.1. First, assume we have a point cloud on the surface $\mathcal{P}^0 = \cup_{i=1}^{n_{\text{surf}}} \{z^{0,i}\}$, where $n_{\text{surf}} \in \mathbb{Z}_{>0}$ is the number of surface points and $z^{0,i} \in \mathbb{R}^2, \forall i \in \mathbb{N}_{1:n_{\text{surf}}}$ are the world Cartesian coordinates of surface points computed with the raw depth readings produced by sensors and current position of the robot. Then, for each surface point $z^{0,i}$, the normal to the surface of the obstacle at that point is approximated by the perpendicular to the segment between the point and the nearest neighbor point within the same scan [92]. Under the assumption that the sensor measurements on the surface are dense enough, this method yields accurate approximations. For each surface point $z^{0,i}$ and corresponding approximated normal to the surface, we construct a set of points

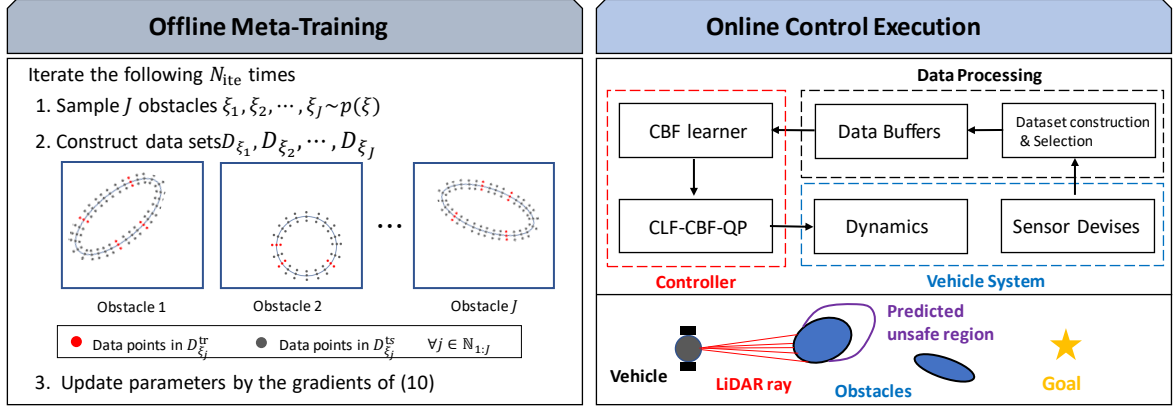


Figure 3.2: The summary of the proposed approach: The proposed method consists of *offline meta-training* and *online control execution*. The offline training follows the procedures in subsection 3.3.2. In each iteration, we first sample J obstacles from the distribution $p(\xi)$ and construct the datasets $D_{\xi_j}^{\text{tr}}$ and $D_{\xi_j}^{\text{ts}}$ for each of these obstacles. Then, the loss (3.10) of the prediction made by the model adopted from $D_{\xi_j}^{\text{tr}}$ is evaluated for test data $D_{\xi_j}^{\text{ts}}$ and the parameters are updated with the gradients of the loss. In the online control execution, the control input is generated by CLF-CBF-QP (3.5) with the CBF candidates obtained through the adaptation of the posterior parameters (3.8) and the probabilistic bounds (3.14). The dataset for the adaptation is constructed using the sensor measurements obtained during the operation by following procedures in subsection 3.3.1 and the data selection scheme discussed in subsection 3.4.1.

$\cup_{p=-n_-}^{n_+} \{z^{p\Delta, i}\}$ with $\Delta \in \mathbb{R}_{>0}$ and $n_+, n_- \in \mathbb{Z}_{\geq 0}$, where $z^{p\Delta, i}, \forall p \in \mathbb{Z}_{-n_-:n_+}$ are the representations of the points that are $p\Delta$ away from the surface of the obstacle (see the red, blue, or green colored points in Fig. 3.1). A negative p value means that the point $z^{p\Delta, i}$ is inside the obstacle. Then, we construct the data set to train h_{IS} as $D = \cup_{i=1}^{n_{\text{surf}}} \cup_{p=-n_-}^{n_+} \{(z^{p\Delta, i}, p\Delta)\}$.

3.3.2 Bayesian meta-learning of the function h_{IS}

When we train the IS function online, the efficiency of the training may be problematic while the model trained offline is not capable of generalizing to new environments. Thus, instead of relying on either online or offline training, we consider employing a meta-learning [86] scheme, which enables quick prediction on h_{IS} with a few amount of online data by effectively using data from different settings. In the following, we first introduce a Bayesian meta-learning framework ALPaCA (Adaptive Learning for Probabilistic Connectionist Architectures) [88]. Then, the probabilistic bounds on the prediction are subsequently discussed.

Overview of ALPaCA

We first parameterize the prediction of the function h_{IS} by the following form:

$$\hat{h}_{\text{IS}}(z; \theta) = \theta^\top \phi_w(z), \quad (3.7)$$

where $\phi_w : \mathbb{R}^2 \rightarrow \mathbb{R}^d$ represents a feed-forward neural network with parameter w and $\theta \in \mathbb{R}^d$ is a coefficient matrix that follows a Gaussian distribution $\mathcal{N}(\bar{\theta}, \sigma^2 \Lambda^{-1})$ which encodes the information and uncertainty associated with the unknown variable $\xi \sim p(\xi)$. Here, $\bar{\theta} \in \mathbb{R}^d$ denotes the mean parameters and $\Lambda \in \mathbb{R}^{d \times d}$ is the positive definite precision matrix. Once the parameters of ϕ_w and the prior distribution, $(w, \bar{\theta}_0, \Lambda_0)$, are determined through the offline meta-training procedure explained later, and the new measurement data along with a fixed $\xi \sim p(\xi)$ is given as $D_\xi = \{(z_i, y_i)\}_{i=1}^{n_{\text{adapt}}}$ with $y_i = h_{\text{IS}}(z_i, \xi)$ by following the procedures in subsection 3.3.1, the mean $\bar{\theta}$ and precision matrix Λ_ξ for the parameter θ are obtained as follows:

$$\Lambda_\xi = \Phi_\xi^\top \Phi_\xi + \Lambda_0, \quad \bar{\theta}_\xi = \Lambda_\xi^{-1} (\Phi_\xi^\top G_\xi + \Lambda_0 \bar{\theta}_0), \quad (3.8)$$

where $G_\xi^\top = [y_1, y_2, \dots, y_{n_{\text{adapt}}}] \in \mathbb{R}^{n_{\text{adapt}}}$, and $\Phi_\xi^\top = [\phi_w(z_1), \phi_w(z_2), \dots, \phi_w(z_{n_{\text{adapt}}})] \in \mathbb{R}^{d \times n_{\text{adapt}}}$. The above computation is based on Bayesian linear regression. Here, the mean and precision matrices are subscripted by ξ to explicitly show that they are calculated using the data associated with unknown variable $\xi \sim p(\xi)$. Then, for any $z \in \mathbb{R}^2$, the mean and variance of the posterior distribution are obtained as

$$\mu_\xi(z) = \bar{\theta}_\xi^\top \phi_w(z), \quad \Sigma_\xi(z) = \sigma^2 (1 + \phi_w(z)^\top \Lambda_\xi^{-1} \phi_w(z)). \quad (3.9)$$

Offline meta-training procedures

In the offline meta-training procedure, the parameters $(w, \bar{\theta}_0, \Lambda_0)$ are trained so that the previously discussed online adaptation yields accurate prediction. To this end, we iterate the following procedures prescribed $N_{\text{ite}} \in \mathbb{Z}_{>0}$ times (the procedures are also summarized in Fig. 3.2). First, we prepare $J \in \mathbb{Z}_{>0}$ data sets by sampling obstacles $\xi_1, \xi_2, \dots, \xi_J$ from the distribution $p(\xi)$ and collect the data for each obstacle as $D_{\xi_j} = \cup_{i=1}^{n_j} \{(z_i^j, y_i^j)\}$, where n_j is the number of data within D_{ξ_j} . Then, we randomly split each dataset D_{ξ_j} into training set $D_{\xi_j}^{\text{tr}} = \cup_{i=1}^{n_j^{\text{tr}}} \{(z_i^{\text{tr},j}, y_i^{\text{tr},j})\}$ and test set $D_{\xi_j}^{\text{ts}} = \cup_{i=1}^{n_j^{\text{ts}}} \{(z_i^{\text{ts},j}, y_i^{\text{ts},j})\}$, where n_j^{tr} and n_j^{ts} are the number of data within $D_{\xi_j}^{\text{tr}}$ and $D_{\xi_j}^{\text{ts}}$ with $n_j^{\text{tr}} + n_j^{\text{ts}} = n_j$ (n_j^{tr} is also randomly chosen from the uniform distribution over $\{1, \dots, n_j\}$ in each iteration). The data within $D_{\xi_j}^{\text{tr}}$ is used to calculate the posterior distribution on the parameter θ_{ξ_j} (i.e., mean $\bar{\theta}_{\xi_j}$ and precision matrix Λ_{ξ_j}) through (3.8) while data within $D_{\xi_j}^{\text{ts}}$ is used to evaluate the following negative log-likelihood loss associated with the prediction based on the above posterior distribution:

$$\begin{aligned}
\ell(\bar{\theta}_0, \Lambda_0, w) &:= - \sum_{j=1}^J \sum_{i=1}^{n_j^{\text{ts}}} \log p(y_i^{\text{ts},j} | z_i^{\text{ts},j}) \\
&= - \sum_{j=1}^J \sum_{i=1}^{n_j^{\text{ts}}} (\log(1 + \phi_w(z_i^{\text{ts},j})^\top \Lambda_{\xi_j}^{-1} \phi_w(z_i^{\text{ts},j})) \\
&\quad - (y_i^{\text{ts},j} - \bar{\theta}_{\xi_j}^\top \phi_w(z_i^{\text{ts},j}))^\top \Sigma_{\xi_j}^{-1} (z_i^{\text{ts},j} \\
&\quad (y_i^{\text{ts},j} - \bar{\theta}_{\xi_j}^\top \phi_w(z_i^{\text{ts},j}))).
\end{aligned} \tag{3.10}$$

Then the parameters are updated with stochastic gradient descent.

Probabilistic bounds on CBF

From the discussions above, we can compute the predictive distribution of the function value $h_{\text{IS}}(z; \xi)$ for any $z \in \mathbb{R}^2$ and $\xi \sim p(\xi)$ by (3.9) once the offline meta-training procedure discussed in subsection 3.3.2 has been done and new measurements are obtained along with the online execution. Here, we consider deriving a deterministic function with probabilistic guarantees, which can readily be incorporated in the QP formulation (3.5). The following discussion follows from [89]. Before deriving the probabilistic bounds on the prediction of h_{IS} , we make two assumptions on the quality of the offline meta-training.

Assumption 7. *For all $\xi \sim p(\xi)$, there exists $\theta_\xi^* \in \mathbb{R}^d$ such that*

$$\theta_\xi^{*\top} \phi_w(z) = h_{\text{IS}}(z; \xi), \quad \forall z \in \mathbb{R}^2. \tag{3.11}$$

Assumption 8. *For all $\xi \sim p(\xi)$ we have the following:*

$$\mathbb{P}(\|\theta_\xi^* - \bar{\theta}_0\|_{\Lambda_t}^2 \leq \sigma^2 \chi_d^2(1 - \delta)) \geq 1 - \delta, \tag{3.12}$$

where $\delta \in \mathbb{R}_{>0}$ is a constant representing a small tolerable probability of failure.

Assumption 7 implies that the meta-learning model (3.7) is capable of fitting the true function while Assumption 8 says that the uncertainty in the prior is conservative enough.

Remark 4. *From the universal approximation theorem [133], Assumption 7 can be satisfied if the meta parameters are appropriately trained. If the condition (3.11) is restrictive in practice, there are some options to relax the condition (3.11) to allow a bounded error as stated in [89].*

Under these assumptions, we can derive the bounds on the function h_{IS} as follows.

Theorem 6. Suppose the offline meta-training of the parameters w , $\bar{\theta}_0$, and Λ_0 is done to satisfy Assumption 7 and 8, and the parameters defining posterior distribution $\bar{\theta}_\xi$ and Λ_ξ are computed by (3.8) with data from the new environment $\xi \sim p(\xi)$. Then, the L_1 norm of the difference between the true function value $h_{\text{IS}}(z; \xi)$ and the mean prediction $\bar{h}_{\text{IS}}(z; \theta_\xi) = \bar{\theta}_\xi^\top \phi(z)$ with $\theta_\xi \sim \mathcal{N}(\bar{\theta}_\xi, \Lambda_\xi)$, for all $z \in \mathbb{R}^2$ and $\xi \sim p(\xi)$, is bounded as the following.

$$|h_{\text{IS}}(z; \xi) - \bar{h}_{\text{IS}}(z; \theta_\xi)| \leq |\phi(z)^\top (\bar{\theta}_\xi - \theta_\xi^*)| \leq \|\phi(z)\|_{\Lambda_\xi^{-1}} \beta_\xi, \quad (3.13)$$

with

$$\beta_\xi = \sigma \left(\sqrt{2 \log \left(\frac{1}{\delta} \frac{\det(\Lambda_\xi)^{1/2}}{\det(\Lambda_0)^{1/2}} \right)} + \sqrt{\frac{\bar{\lambda}(\Lambda_0)}{\underline{\lambda}(\Lambda_\xi)} \chi_d^2(1 - \delta)} \right),$$

with probability at least $(1 - 2\delta)$.

Proof. Proof of this result follows from [89] and [134]. In the middle of the proof of Theorem 1 in [89], it is shown that $|a^\top (\bar{\theta}_\xi - \theta_\xi^*)| \leq \|a\|_{\Lambda_\xi^{-1}} \beta_\xi$ holds for any $a \in \mathbb{R}^d$ with probability at least $(1 - 2\delta)$ (see pp. 18 of [89]). Then, (3.13) is obtained by substituting $\phi_w(z)$ to a in the inequality. $\square$

The bound in Theorem 6 is derived by adapting the result from the context bandit literature [134]. The expression on the right-hand side of (3.13) approaches zero when the number of data points tends to infinity since the regret bound in [134] achieves a sub-linear rate.

Since the difference between the true value and mean prediction is probabilistically bounded by $\|a\|_{\Lambda_\xi^{-1}} \beta_\xi$ as discussed in Theorem 6, we can provide the associated lower bound of h_{IS} corresponding to a robot state x and obstacle parameter ξ as the following.

$$h^b(x; \theta_\xi) := \bar{h}_{\text{IS}}(v(x); \theta_\xi) - \|\phi(v(x))\|_{\Lambda_\xi^{-1}} \beta_\xi, \quad (3.14)$$

where $v : \mathbb{R}^n \rightarrow \mathbb{R}^2$ is the function that maps the robot's state to the robot's position in 2D space. If $h^b(x; \theta_{\xi_i}) > 0$ holds for all $i \in \mathbb{N}_{1:N_{\text{obs}}}$ at a point x , then the probability of the point $v(x)$ being outside of all the N_{obs} obstacles is obtained as $(1 - 2\delta)^{N_{\text{obs}}}$ from Theorem 6. By using h^b as the CBF, we can provide a probabilistic guarantee of the resulting controller as elaborated in Section 3.4.

Remark 5. In the GPIS training discussed in this section, the dataset basically relies on points near the obstacle surfaces and this can affect the conservativeness of the model's uncertainty prediction in free space (especially locations far from the obstacle). We can partially address this issue by setting n_+ defined in subsection 3.3.1 large value depending on the learning problem. More radical solutions would be considered in future work.

Remark 6. Although our proposed method directly models the distance field as a GP similar to most standard GPIS techniques, enforcing the Eikonal property in training as proposed in [135] may increase the model's efficiency and accuracy, especially for complex environments.

Algorithm 2: Online control execution

Input : f, g (system dynamics model); $p(\xi)$ (distribution of obstacles); $(w, \bar{\theta}_0, \Lambda_0)$ (prior parameters); δ (confidence level); V (CLF); T (execution time)

```
1  $\xi_i \sim p(\xi), \mathcal{B}_i \leftarrow \emptyset, \forall i \in \mathbb{N}_{1:N_{\text{obs}}};$   
2 while  $t < T$  do  
3   if  $t = k\Delta_{\text{lidar}}, k \in \mathbb{Z}_{\geq 0}$  then  
4     [CBF candidate update]  
5     for  $i = 1 : N_{\text{obs}}$  do  
6       Obtain sensor measurements  $\mathcal{P}_i^0$ ;  
7       Construct dataset  $D_i$  by procedures in subsection 3.3.1  
8       Update the buffer  $\mathcal{B}_i$  by procedures in subsection 3.4.1;  
9       if  $\mathcal{B}_i \neq \emptyset$  then  
10        Compute the posterior parameters  $\bar{\theta}_{\xi_i}$  and  $\Lambda_{\xi_i}$  by (3.8);  
11        Set  $h^b(\cdot; \theta_{\xi_i}) = h_{IS}^b(v(\cdot); \theta_{\xi_i})$  by (3.14);  
12      end  
13    end  
14  end  
15  [Control execution]  
16  Solve the QP (3.5) with the CBF candidates above;  
17  Solve (3.1) and update state  $x(t)$ ;  
18 end
```

3.4 Online Control Execution

Given system (3.1), meta-learned parameters $(w, \bar{\theta}_0, \Lambda_0)$, and CLF V that encodes a control objective, the procedures in the online control execution is as summarized in Algorithm 2. Before the execution, an environment is determined by sampling the obstacles $\xi_i, \forall i \in \mathbb{N}_{1:N_{\text{obs}}}$ from the distribution $p(\xi)$ and the data buffers $\mathcal{B}_i$ for $\xi_i, \forall i \in \mathbb{N}_{1:N_{\text{obs}}}$ are initialized by empty sets (Line 1). Then, the control is executed by repeatedly solving the QP (3.5) with the learned functions (3.14) defined using the current posteriors $\bar{\theta}_{\xi_i}$ and Λ_{ξ_i} calculated by the data in the current buffers (Line 10). The buffers $\mathcal{B}_i$ and the posterior parameters $\bar{\theta}_{\xi_i}$ and $\Lambda_{\xi_i} \forall i \in \mathbb{N}_{1:N_{\text{obs}}}$ are updated at every $\Delta_{\text{lidar}} \in \mathbb{R}_{>0}$ [s] through the procedures in the following subsection (Line 3-14). Since each buffer $\mathcal{B}_i$ is empty at the beginning, we ignore the CBF constraint for ξ_i until surface points of the corresponding obstacle are detected. To justify this, we make the following assumption.

Assumption 9. *The length of the laser is long enough and the measurements are taken constant enough such that the control invariant of the safe region is maintained even if the unobserved obstacles are*

ignored in the current QP.

3.4.1 The update scheme of each buffer $\mathcal{B}_i$

The update procedures of each buffer $\mathcal{B}_i$ are discussed here. We ignore the subscription i in the following discussion for simplicity of notation. At time instance k , suppose we have new data points $D_k = \cup_{i=1}^{n_k} \cup_{p=n_-}^{n_+} \{(z_k^{+p\Delta, i}, p\Delta)\}$ obtained by following the procedures in subsection 3.3.1 and the buffer $\mathcal{B}$ which is constructed by the data points obtained at past time instances 0 to $k-1$. Intuitively, the data points in D_k will include many data points that are similar to those obtained in the past instances if the time interval of updating the dataset is small. Thus, it is important to select only the informative data points before adding them to the buffer. Here, we consider the data selection scheme for this purpose based on [136]. We repeat the following procedures for $i \in \mathbb{N}_{1:n_k}$. First, we compute the predictive variance at a data point on the surface $z_k^{0,i}$ with posterior parameters calculated by the current $\mathcal{B}$ as $\Sigma_\xi(z_k^{0,i})$. Then, if the value $\Sigma_\xi(z_k^{0,i})$ is larger than the prescribed threshold $\eta \in \mathbb{R}_{\geq 0}$, the data points $\cup_{p=n_-}^{n_+} (z_k^{p\Delta, i}, p\Delta)$ are considered to be informative for the current model and thus added to the buffer $\mathcal{B}$.

3.4.2 Theoretical results

The probabilistic safety guarantees for our control scheme are summarized in the following theorem.

Theorem 7. *Suppose Assumption 9 holds and the offline meta-training of the parameters w , $\bar{\theta}_0$, and Λ_0 is performed to satisfy Assumption 7 and 8, and feasible solution of the QP in Algorithm 2 is obtained at all time $t < T$. Then, $x(t) \in C_{\xi_i}$ holds with probability at least $1 - 2\delta$ for all $t \in \mathbb{R}_{>0}$ and $i \in \mathbb{N}_{1:N_{\text{obs}}}$.*

Proof. From Theorem 4, the implicit control policy defined by the QP in Algorithm 2 keeps the system states within $\hat{C}_{\xi_i} := \{x \in \mathcal{D} \subset \mathbb{R}^n \mid h^b(x, \xi_i) \geq 0\}$ for all of the observed obstacles ξ_i . Since each state in $\hat{C}_{\xi_i}$ is included in the actual safe set C_{ξ_i} with probability at least $1 - 2\delta$ from Theorem 6, we can guarantee $x(t) \in C_{\xi_i}$ with probability at least $1 - 2\delta$ for each $t \in \mathbb{R}_{>0}$ and the observed obstacles ξ_i . This result and Assumption 9 prove the theorem. $\square$

Note that the safety guarantee in Theorem 7 is point-wise rather than trajectory-wise similar to the previous works such as [98, 99, 137] since the prediction of the boundary on the safe region is determined from the probabilistic bound (3.14).

3.5 Case Study

We test the proposed approach through a simulation of vehicle navigation in a 2D space. All the experiments are conducted on Python running on a Windows 10 with a 2.80 GHz Core i7 CPU and 32 GB of RAM. TensorFlow and PyBullet [138] are used in the implementation of the meta-learning and construction of the simulation environment, respectively. We first define the controlled system by the unmanned vehicle that has the following dynamics:

$$\underbrace{\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{\vartheta} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} \cos \vartheta & -\ell \sin \vartheta \\ \sin \vartheta & \ell \cos \vartheta \\ 0 & 1 \end{bmatrix}}_{g(x)} \underbrace{\begin{bmatrix} v \\ \omega \end{bmatrix}}_u, \quad (3.15)$$

where q_x , q_y , and ϑ are $x - y$ coordinates of the robot position and heading angle of the robot, v and ω are the velocity and angular velocity of the vehicle, respectively. The control inputs and system states are defined as $u = [v, \omega]^\top \in \mathbb{R}^2$ and $x = [q_x, q_y, \vartheta]^\top \in \mathbb{R}^3$, respectively. Note that in (3.15), we consider the dynamics of a point off the axis of the robot by a distance ℓ (> 0) to make the system relative degree one. Moreover, the LiDAR scanner which has a 360-degree field of view and radiates 150 rays with a 3-meter length is assumed to be mounted on the vehicle. In the following subsections, we compare the proposed method and the case where we use GP [51], a commonly used Bayesian method, instead of ALPaCA. We choose a zero mean GP with a squared exponential kernel and the GP is implemented by the Python library GPy.

3.5.1 Meta learning of unsafe regions

We first test the performance of the proposed scheme for constructing the IS function $h_{\text{IS}} : \mathbb{R}^2 \rightarrow \mathbb{R}$. We consider 2 settings: (I) Ellipsoidal obstacles with randomly chosen semi-axes lengths, center positions, and rotation angles (II) Objects in Fig. 3.3 with randomly chosen positions, rotation angles, and scales. In setting (I), the semi-axes lengths along the x and y coordinates, rotation angles, and centers of ellipsoids are randomly sampled from the uniform distribution in the range $[0.4, 0.8]$, $[-0.8, 0.8]$, and $[0, 2\pi]$, respectively. In setting (II), we used PyBullet objects to test with more arbitrary shapes. The rotational angles and the center position of the objects are sampled in the same way as setting (I). For both cases, the architecture of ϕ in (3.7) is defined by three hidden layers of 256 units with 32 basis functions, and meta-parameters are trained with the procedure in subsection 3.3.2. Moreover, we set N_{ite} , σ , n_- , n_+ , and Δ to 30000, 0.001, 1, 5, and 0.1, respectively. A few examples of the prediction of obstacle surfaces calculated with small amounts of data are shown in

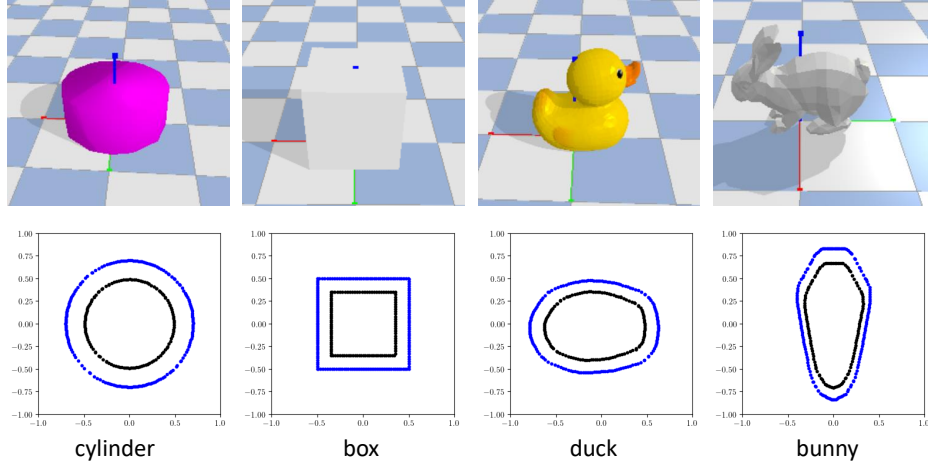


Figure 3.3: PyBullet objects in case (II). The figures in the first and second lows are the GUI of the objects and corresponding cross sections at x - y coordinates, respectively. The surfaces with the smallest and largest sizes are shown by the black and blue lines, respectively.

Table 3.1: The CSEs between the robot position and the goal position

Method \ Env	Env					
	1	2	3	4	5	6
Proposed	619	830	654	843	487	1077
GP case	1011	1095	897	1804	503	1648

Fig. 3.4. The blue lines and blue shaded areas in the figure are the mean predictions of the surfaces and the regions between the 0-level sets of the upper and lower 90% bounds of h_{IS} . We can see that the confidence bounds are reasonably tight even when the number of data points is small, which is the main effect of the offline meta-training. Similar results are also observed for different obstacles. The average negative log-likelihood (NLL) for cases (I) and (II), and the time required for adaptation are also summarized in Fig. 3.5. We can see that the prediction made by the proposed scheme is superior to that of the GP case when the number of data is small. Moreover, same as reported in [88], the meta-learning scheme can produce the prediction very fast compared to the GP case.

3.5.2 Control execution

Based on the trained h_{IS} , we next consider executing the control with Algorithm 2. The control objective is to steer the vehicle toward the goal position (q_x^g, q_y^g) while avoiding obstacles. The goal-reaching objective is encoded through the CLF $V(x) = (q_x - q_x^g)^2 + (q_y - q_y^g)^2$. Moreover, α_C and

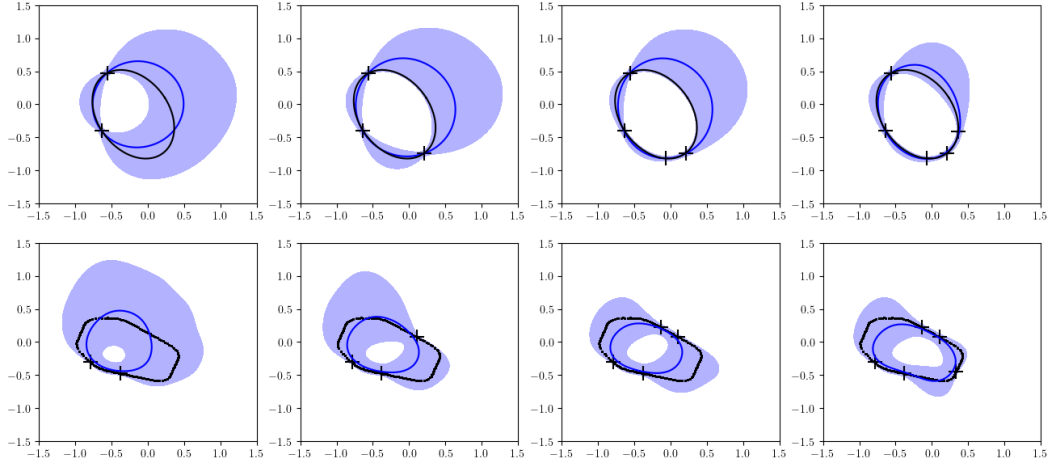


Figure 3.4: A few examples of the prediction on surfaces: The black and blue lines show the actual surface and the mean prediction of the surface, respectively. The blue shaded areas show the regions between the 0-level sets of the upper and lower 90% bounds of h_{IS} obtained from the discussion in subsection 3.3.2, and the black cross marks show the data points on the surface used for the adaptation.

α_V in (3.5) are defined by the linear functions $\alpha_C(c) = \gamma_C c$ and $\alpha_V(c) = \gamma_V c$. We set parameters γ_C , γ_V , λ , and η to 1.0, 1.0, 10, and 0.003 respectively. Since the theoretically guaranteed bounds of GP [52, 139] tend to be overly conservative, we instead use the $2\text{-}\sigma$ lower bounds in the GP case. The metric of the control performance is defined by the cumulative squared error (CSE) of the difference between the robot and goal positions, along with the resulting state trajectory (the squared errors are corrected every 0.02 [s]). The experiment is conducted on 6 different environments shown in Fig. 3.6. An example of the control execution in Env 1 with $\Delta_{\text{lidar}} = 5$ [s] is visualized in Fig. 3.7. From Fig. 3.7, we can see the vehicle successfully reaches the goal while avoiding the unsafe region. Notably, the control of the proposed method is much less conservative than the GP case because of the tight prediction of the unsafe region (the vehicle in the GP case is often forced to be stacked in small safe regions until the subsequent update of the CBF). The CSEs when $\Delta_{\text{lidar}} = 5$ for all 6 environments are summarized in Table 3.1 and we can confirm the less conservative performance of the proposed scheme for all the environments. Although the CSEs of the proposed method and the GP case get closer as Δ_{lidar} becomes smaller, it is ideal to maintain the value Δ_{lidar} to be large to save computation and the use of sensor devices that have limited batteries. Moreover, the GP can not be updated so frequently because of the time required for the training (see Fig. 3.5).

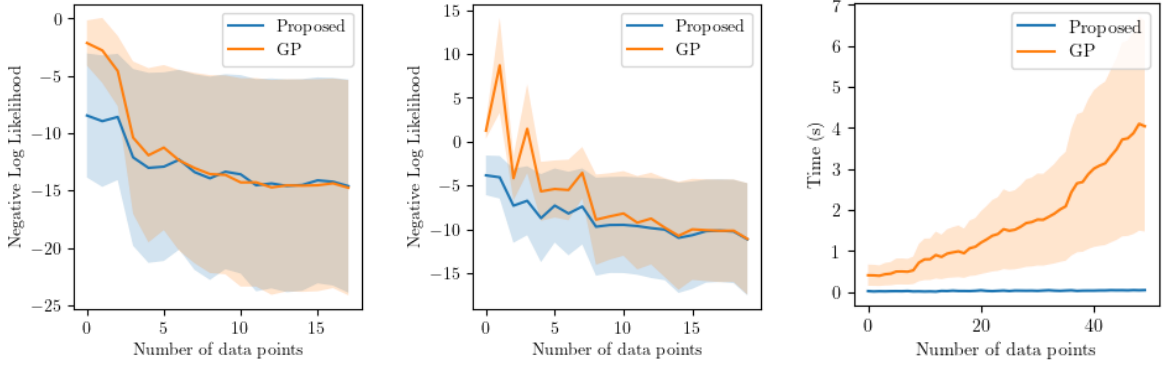


Figure 3.5: **Left and Middle:** NLLs for predictions of the proposed meta-learning scheme and GP case against the number of data points for the cases (I) and (II) (the number of data points in the figure means the number of surface points and the actual number of data points is $n_+ + n_- + 1 (= 7)$ times of that). **Right:** the time required for the training against the number of data points. The means and $3 - \sigma$ intervals are plotted in the figures.

3.6 Summary

In this study, we proposed an efficient and probabilistically ensured online learning method of CBF with sensor measurements. In the proposed method, we utilized a technique based on the GPIS to train unsafe regions from the measurements. Specifically, a Bayesian meta-learning scheme was employed to learn IS function, which enables us to effectively use past data from different settings. To avoid the volume of data buffers becoming unnecessarily large in the online control execution, we also considered the data selection scheme which effectively uses the uncertainty information provided by the current model. The prediction made by the proposed scheme was readily incorporated in the CBF-CLF-QP formulation and the probabilistic safety guarantee for the control scheme was also provided. In the case study, we have shown the efficacy of our method in terms of the conservativeness of the prediction, the time required for the online training, and the conservativeness of the control performance.

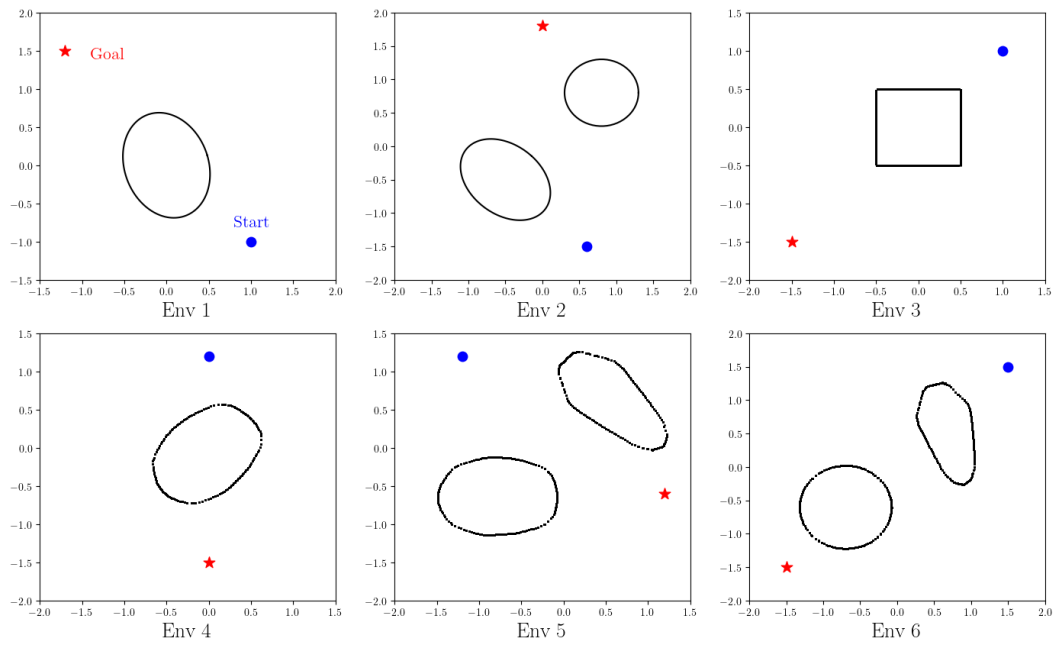


Figure 3.6: Environments for testing the control performance.

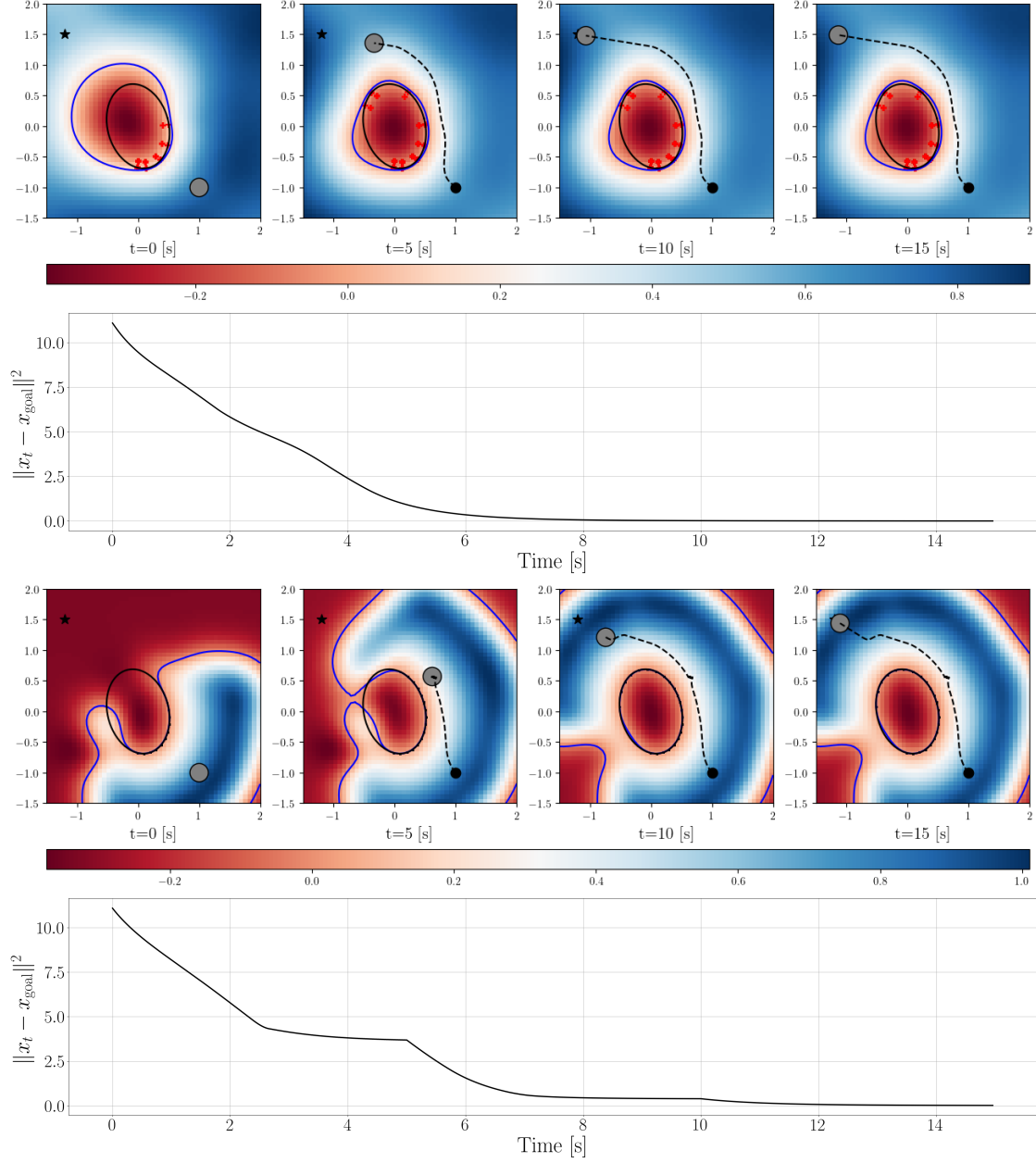


Figure 3.7: The control execution in Env 1 with the proposed method (above) and GP case (below). The red crosses, heat maps, blue solid lines, black solid lines, and black dotted lines represent data points on the surface, the values of CBF, 0-level sets of 95% lower bounds of CBF, actual surfaces, and robot trajectories, respectively. The CBF is updated every 5 [s].

Chapter 4

STL2vec: Signal Temporal Logic Embeddings for Control Synthesis With Recurrent Neural Networks

In this chapter, a method for learning a recurrent neural network (RNN) controller that maximizes the robustness of signal temporal logic (STL) specifications is presented. In contrast to previous methods, we consider synthesizing the RNN controller for which the user is able to select an STL specification arbitrarily from multiple STL specifications. To obtain such a controller, we propose a novel notion called STL2vec, which represents a vector representation of the STL specifications and exhibits their similarities. The construction of the STL2vec is useful since it allows us to enhance the efficiency and performance of the controller. We validate our proposed method through the examples of the path planning problem.

This chapter is organized based on “STL2vec: Signal Temporal Logic Embeddings for Control Synthesis With Recurrent Neural Networks” [140] which appeared in IEEE Robotics and Automation Letters.

4.1 Preliminaries

4.1.1 System description and notations

We consider a non-linear discrete-time system of the form:

$$x_{t+1} = f(x_t, u_t), \quad x_0 \in \mathcal{X}_0, \quad u_t \in \mathcal{U}, \quad (4.1)$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathcal{U}$ are the state and the control input at time $t \in \mathbb{Z}_{\geq 0}$, $\mathcal{X}_0 \subset \mathbb{R}^n$ is the set of initial states, $\mathcal{U} \subset \mathbb{R}^m$ is the set of control inputs, and $f : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^n$ is a function capturing the dynamics of the system. We assume that the initial state x_0 is randomly chosen from $\mathcal{X}_0$ according to the probability distribution $p : \mathcal{X}_0 \rightarrow \mathbb{R}$, and $\mathcal{U}$ is given by $\mathcal{U} = \{u \in \mathbb{R}^m : u_{\min} \leq u \leq u_{\max}\}$ for given $u_{\max}, u_{\min} \in \mathbb{R}^m$ (the inequalities are interpreted element-wise). Given $x_0 \in \mathcal{X}_0$ and a sequence of control inputs $u_0, \dots, u_{T-1}$, we can generate a unique sequence of states according to the dynamics (4.1), which we call a *trajectory*: $x_{0:T} = x_0, x_1, \dots, x_T$.

4.1.2 Signal Temporal Logic

Signal temporal logic (STL) [37] is a logical formalism that can specify temporal properties of real-valued signals. The syntax of the STL formula is recursively defined as follows:

$$\begin{aligned} \phi := & \top \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \mathbf{F}_I\phi \mid \\ & \mathbf{G}_I\phi \mid \phi_1 \mathbf{U}_I\phi_2 \end{aligned} \quad (4.2)$$

where $\mu : \mathbb{R}^n \rightarrow \mathbb{B}$ is the predicate whose boolean truth value is determined by the sign of a function $h : \mathbb{R}^n \rightarrow \mathbb{R}$, $\top$, $\neg$, $\wedge$, and $\vee$ are Boolean *true*, *negation*, *and*, and *or* operators, respectively, and $\mathbf{F}_I$, $\mathbf{G}_I$, $\mathbf{U}_I$ are the temporal *eventually*, *always*, and *until* operators defined on a time interval $I = [a, b] = \{t \in \mathbb{Z}_{\geq 0} : a \leq t \leq b\}$ ($a, b \in \mathbb{Z}_{\geq 0}$).

We define the semantics of an STL formula ϕ with respect to the trajectory $\mathbf{x} := x_{0:T}$ of the system (4.1) at time t as follows:

$$\begin{aligned} (\mathbf{x}, t) \models \mu & \Leftrightarrow h(x_t) > 0 \\ (\mathbf{x}, t) \models \neg\mu & \Leftrightarrow \neg((\mathbf{x}, t) \models \mu) \\ (\mathbf{x}, t) \models \phi_1 \wedge \phi_2 & \Leftrightarrow (\mathbf{x}, t) \models \phi_1 \wedge (\mathbf{x}, t) \models \phi_2 \\ (\mathbf{x}, t) \models \phi_1 \vee \phi_2 & \Leftrightarrow (\mathbf{x}, t) \models \phi_1 \vee (\mathbf{x}, t) \models \phi_2 \\ (\mathbf{x}, t) \models \mathbf{F}_I\phi & \Leftrightarrow \exists t_1 \in t + I, (\mathbf{x}, t_1) \models \phi \\ (\mathbf{x}, t) \models \mathbf{G}_I\phi & \Leftrightarrow \forall t_1 \in t + I, (\mathbf{x}, t_1) \models \phi \\ (\mathbf{x}, t) \models \phi_1 \mathbf{U}_I\phi_2 & \Leftrightarrow \exists t_1 \in t + I \text{ s.t. } (\mathbf{x}, t_1) \models \phi_2 \\ & \wedge \forall t_2 \in [t, t_1], (\mathbf{x}, t_2) \models \phi_1, \end{aligned}$$

where $t + I = \{t + k \in \mathbb{Z}_{\geq 0} : k \in I\}$. For simplicity, we denote $\mathbf{x} \models \phi$ to abbreviate $(\mathbf{x}, 0) \models \phi$. The above semantics is *qualitative*, in the sense that it reveals only if the trajectory $\mathbf{x}$ either satisfies or violates ϕ .

The notion of *robustness* of STL formulas provides *quantitative* semantics, and it measures how much the trajectory satisfies the STL formula. The robustness is sound in the sense that positive robustness value implies satisfaction of STL formula and negative robustness implies violation of STL formula. The robustness score of the STL formula ϕ is defined with respect to a trajectory $\mathbf{x}$ and a time t , which we denote by $\rho^\phi(\mathbf{x}, t)$, and is recursively defined as follows:

$$\begin{aligned}
\rho^\mu(\mathbf{x}, t) &= h(x_t) \\
\rho^{\neg\mu}(\mathbf{x}, t) &= -h(x_t) \\
\rho^{\phi_1 \wedge \phi_2}(\mathbf{x}, t) &= \min(\rho^{\phi_1}(\mathbf{x}, t), \rho^{\phi_2}(\mathbf{x}, t)) \\
\rho^{\phi_1 \vee \phi_2}(\mathbf{x}, t) &= \max(\rho^{\phi_1}(\mathbf{x}, t), \rho^{\phi_2}(\mathbf{x}, t)) \\
\rho^{F_I \phi}(\mathbf{x}, t) &= \max_{t_1 \in t+I} \rho^\phi(\mathbf{x}, t_1) \\
\rho^{G_I \phi}(\mathbf{x}, t) &= \min_{t_1 \in t+I} \rho^\phi(\mathbf{x}, t_1) \\
\rho^{\phi_1 U_I \phi_2}(\mathbf{x}, t) &= \max_{t_1 \in t+I} \left(\min(\rho^{\phi_2}(\mathbf{x}, t_1), \right. \\
&\quad \left. \min_{t_2 \in [t, t_1]} \rho^{\phi_1}(\mathbf{x}, t_2)) \right)
\end{aligned}$$

Note that the trajectory length T should be selected large enough to determine the robustness score (see, e.g., [40]). For simplicity, we denote $\rho^\phi(\mathbf{x})$ to abbreviate $\rho^\phi(\mathbf{x}, 0)$.

4.2 Problem Statement

Let us now formulate a problem that we seek to solve throughout the chapter. First, let $\Phi = \{\phi_1, \phi_2, \dots, \phi_M\}$ denote a set of M *candidate* STL specifications. We assume that the STL specification can be freely chosen from the M candidate specifications by the user before the control execution. Once the STL specification is chosen by the user, say $\phi_i \in \Phi$, the system (4.1) is then controlled aiming to satisfy ϕ_i . We also assume for simplicity that the chosen STL specification is fixed during control execution (as detailed below).

In this work, we aim to learn a feedback controller that maximizes the robustness of the STL specification. Note that the satisfaction and the robustness of an STL specification are defined over the trajectory of the system (4.1), and that the STL specification is freely chosen from Φ . Hence, we seek a parameterized control policy $\pi(\cdot; W)$ that maps the state history and the STL specification to the control input:

$$u_t = \pi(x_{0:t}, \phi_i; W), \quad x_{0:t} := (x_0, \dots, x_t), \quad \phi_i \in \Phi,$$

where $x_{0:t}$ is the state trajectory generated by (4.1), ϕ_i is an STL specification selected from Φ , and

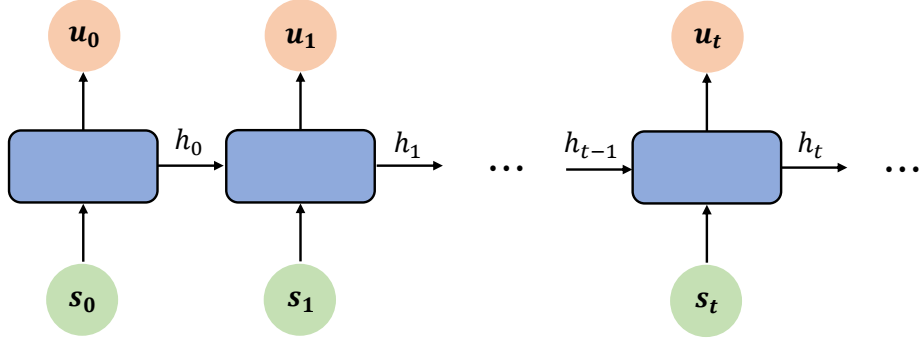


Figure 4.1: The structure of RNN.

W denotes the learnable parameters of the policy π . More formally, the problem considered in this chapter is defined as follows:

Problem 1. *Given the system (4.1), horizon length T , probability distribution of initial states $p : \mathcal{X}_0 \rightarrow \mathbb{R}$, and the candidate STL specifications $\Phi = \{\phi_1, \phi_2, \dots, \phi_M\}$, find a set of parameters W that is the solution to the following problem:*

$$\underset{W}{\text{maximize}} \quad \frac{1}{M} \sum_{i=1}^M \mathbb{E}_{p(x_0^i)} [\rho^{\phi_i}(x_{0:T}^i)] \quad (4.3)$$

$$\text{s.t. } x_{t+1}^i = f(x_t^i, \pi(x_{0:t}^i, \phi_i; W)), \quad (4.4)$$

$$t = 0, 1, \dots, T-1, i = 0, 1, \dots, M,$$

where $\pi(\cdot, \cdot; W)$ denotes a control policy parameterized by W and $x_{0:t}^i = x_0^i, x_1^i, \dots, x_t^i$ is the trajectory of (4.1) along with the policy $\pi(\cdot, \phi_i; W)$. $\square$

In Problem 1, we look for a set of the parameters of the control policy W that maximizes the sum of the expectation of the robustness score with respect to the distribution of initial states over all M candidate STL specifications. Since the control policy depends on the past and present states, or it is *history dependent*, in this study we use a recurrent neural network (RNN) to learn the control policy π (see, [43, 44]). RNN is a type of a neural network that has a feedback architecture. A basic structure of the RNN is illustrated in Fig. 4.1. As shown in Fig. 4.1, RNN keeps processing the sequential information via internal hidden state h . The update rule of the hidden state and derivation of the output at time t are as follows:

$$h_t = g_{W_1}(h_{t-1}, s_t), u_t = l_{W_2}(h_t), \quad (4.5)$$

where g_{W_1} and l_{W_2} are the functions parameterized by weights W_1 and W_2 , respectively, and s_t denotes the input that is fed to the RNN. Hence, the set of the RNN parameters is given by $\{W_1, W_2\}$. As

discussed in [45], we can restrict the output of the RNN (i.e., the control input u_t) within the lower bound $u_{\min}$ and the upper bound $u_{\max}$ by defining each element i ($i = 1, \dots, m$) of the function l_{W_2} as follows:

$$u_{t,i} = u_{\min,i} + \frac{u_{\max,i} - u_{\min,i}}{2} (\tanh([W_2 h_t]_i) + 1), \quad (4.6)$$

where the subscript i denote i -th element of the vector. Using (4.6), we can generate control inputs satisfying $u_t \in \mathcal{U}$. A detailed definition of s_t in (4.5) as well as a concrete procedure to learn W_1, W_2 are elaborated in Section 4.4.

Indeed, solving Problem 1 is not trivial and challenging in the following sense. Since the control policy depends on an STL specification ϕ_i , we should consider how to transform each STL specification $\phi_i \in \Phi$ into a corresponding *vector*, so that it can be fed to the RNN as the inputs together with the state trajectory $x_{1:t}$. Intuitively, it may be desirable that we can provide similar inputs to the RNN if the two STL specifications are close to each other in terms of their control policies. For example, consider a simple case with $x \in \mathbb{R}$ and the STL candidate specifications

$$\phi_1 = \mathbf{F}_{[0,10]}(0 \leq x \leq 1), \quad (4.7)$$

$$\phi_2 = \mathbf{F}_{[0,11]}(0 \leq x \leq 1), \quad (4.8)$$

$$\phi_3 = \mathbf{F}_{[0,10]}(10 \leq x \leq 11), \quad (4.9)$$

$$\phi_4 = \mathbf{F}_{[0,10]}(10 \leq x \leq 11) \vee \mathbf{F}_{[0,10]}(12 \leq x \leq 13). \quad (4.10)$$

Intuitively, control policies to satisfy ϕ_1 and ϕ_2 may be almost the same (as we aim to control the system to the same region with almost the same time intervals), while the control policies to satisfy ϕ_1 and ϕ_3 may be quite different (as we aim to control the system to different regions). In addition, if we could find a control policy to satisfy ϕ_3 , this control policy also leads to the satisfaction of ϕ_4 . Hence, it is convenient that we could provide a certain *mapping*, where ϕ_1 and ϕ_2 are mapped onto the vector points that are close to each other but far from those corresponding to ϕ_3 and ϕ_4 . In addition, ϕ_3 and ϕ_4 are mapped onto the same vector points if a control policy to satisfy ϕ_3 could be found.

Motivated by the above intuition, in this study, we propose a novel scheme for constructing a *vector representation* of the STL specifications, which is referred to as *STL2vec*. The proposed approach is inspired by the notion of Word2vec [121], a vector representation of words that has been proposed in machine learning literature and widely utilized for natural language processing. A concrete procedure of the proposed approach is elaborated in Section 4.4.

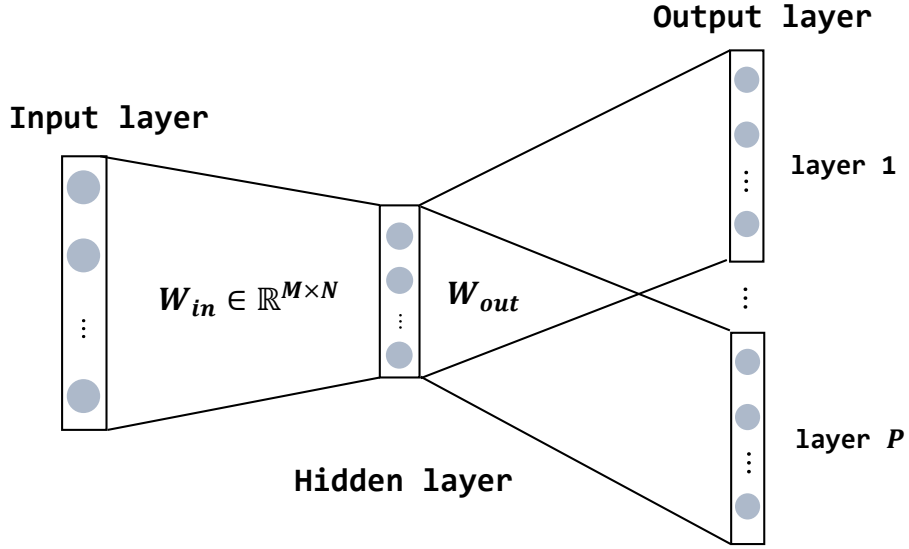


Figure 4.2: Skip-gram model.

4.3 Summary of Word2vec (skip-gram)

Before providing the proposed approach, we summarize the basic concept of Word2vec. As we will see in the next section, the concept of Word2vec is a key ingredient to introduce STL2vec and how to train for it. The main objective of Word2vec is to group the vectors of similar or related words (for example, the words "man" and "boy" are mapped onto similar points in the vector space). As such, we can let the computer perform mathematical operations on words to detect their similarities. The mapping from each word to the vector is represented via an NN, i.e., the input for the NN is a word, and its output is a corresponding vector. One of the common techniques to train the NN is the *skip-gram*[121]. The structure of the skip-gram is shown in Fig.4.2. As shown in Fig.4.2, the skip-gram has a shallow three-layer NN. The input to the NN is an M -dimensional one-hot vector, i.e., only one single element is 1 and the other elements are all zero, where M indicates a total number of words in the corpus (typically with $M > 10^6$). Here, each word is assigned by a unique one-hot vector (for example, "man" is labeled by $[1, 0, 0, \dots, 0]^T$, and "boy" is labeled by $[0, 1, 0, \dots, 0]^T$, etc.), so that each word in the corpus is readable to the NN. The outputs of the NN are the collection of P (M -dimensional) vectors whose each element represents the probability of the corresponding word in the corpus (training data for the output layer will be explained later in this section), where P is a user-defined parameter. The mapping from the input layer to the hidden layer is given by a matrix $W_{in} \in \mathbb{R}^{M \times N}$, where N is a user-defined parameter that indicates the dimension of the word vector. The mapping from the hidden layer to each output layer is given by another matrix $W_{out} \in \mathbb{R}^{N \times M}$.

Here, we note that the matrix W_{out} is the same for all the output layers. The weight parameters $W_{\text{in}}, W_{\text{out}}$ are trained from a large number of sentences, such as those in the news report (see, e.g. [121]). For example, suppose $P = 2$ and we have a sentence: "I want to eat an orange every morning". Then, we regard the training input as a certain word in this sentence and the training output as the set of $P = 2$ words around it. For example, the word "eat" is regarded as the training input, and the set of $P = 2$ words around it, i.e., {to, an} is regarded as the training outputs. Generating the training dataset as above is motivated by the insight that the meaning of a word is established by the surrounding words.

Using the dataset constructed above, we train the skip-gram model. First, we pass the one-hot encoded input data to the input layer and perform the forward computation. As an output layer and loss function, we use softmax and cross-entropy loss, respectively. Then, we update the weights W_{in} and W_{out} using the gradients obtained by back-propagation computation (for details, see [121]). After the training, the vector representation of each word in the corpus is given through the projection of the weight matrix W_{in} (hence, we drop the matrix W_{out} in the skip-gram model). That is, for each word w in the corpus, the corresponding vector representation is given by $z_w = W_{\text{in}}e_w \in \mathbb{R}^N$, where $e_w \in \mathbb{R}^M$ denotes the one-hot vector of the word w .

4.4 Proposed Scheme

In this section, we describe the solution approach to Problem 1. The proposed approach consists of the two steps: (i) train a vector representation of the STL specifications, namely STL2vec, whose input is the STL specification $\phi_i \in \Phi$ and output is the N -dimensional vector; (ii) train the RNN, whose inputs are the state trajectory $x_{0:t}$ and the vector representation of the STL specification z_{ϕ_i} , and the output is the control input u_t . The illustration of the two steps are shown in Fig. 4.3. Since STL2vec will be trained based on the skip-gram, the parameter to be learned in Step (i) is W_{in} (recall in Section 4.3 that we neglect W_{out}). Also, recall that the parameters to be learned for RNN in Step (ii) is $\{W_1, W_2\}$ (see Section 4.2). Hence, the overall parameters to be learned for the control policy π is $W = \{W_{\text{in}}, W_1, W_2\}$.

The proposed approach is beneficial in the following sense. By constructing STL2vec, we can obtain a vector representation that exhibits similarities between the STL specifications. This allows us to accelerate both efficiency and performance of the RNN controller in contrast to some naive approaches, such as integer or one-hot encoding, i.e., simply assigning arbitrary numbers or one-hot vectors to the STL specifications (e.g., ϕ_1 is assigned by 1, ϕ_2 is assigned by 2, and so on) and use these numbers as the inputs to the RNN; for details, see an experimental result in Section 4.5. Moreover, the proposed approach is more beneficial than the approach of learning RNN controllers one by one for

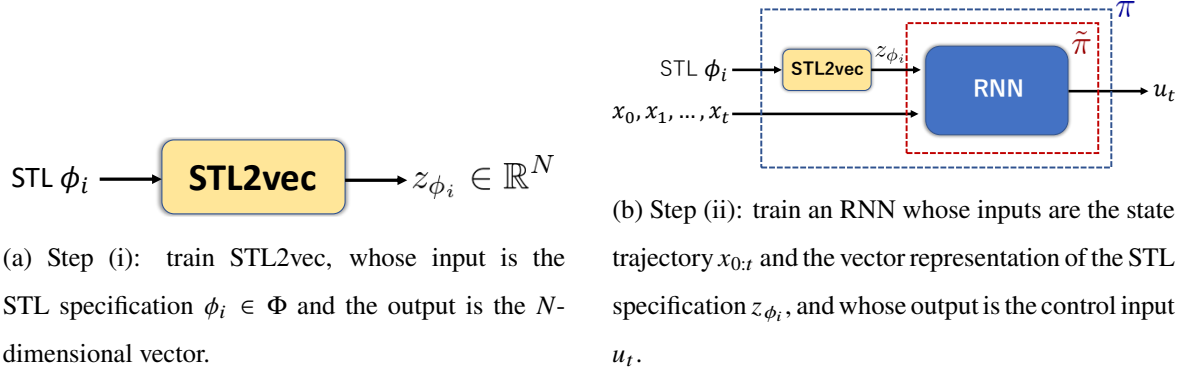


Figure 4.3: Overview of the proposed approach.

each STL formula in terms of computational memory; for details, see subsection 4.4.3 and Section 4.5. The concrete procedures of the above two steps are given in the following subsections.

4.4.1 Training STL2vec

In this subsection, we provide a detailed procedure of Step (i) (train STL2vec). To train STL2vec, we use the skip-gram model as explained in Section 4.3. First, we encode all the STL candidate specifications by the M -dimensional one-hot vectors, i.e., each STL specification $\phi_i \in \Phi$ is encoded by an M -dimensional vector whose i -th element is 1, and all other elements are 0 (ϕ_1 is encoded as $[1, 0, \dots, 0]^T$ and ϕ_2 is encoded as $[0, 1, 0, \dots, 0]^T$, and so on), so that they are readable to the NN. For simplicity, the one-hot vector of $\phi_i \in \Phi$ is denoted by $e_{\phi_i} \in \{0, 1\}^M$.

A key question regarding the construction of STL2vec is how to learn the weight parameters $W_{\text{in}}, W_{\text{out}}$ for the skip gram model, or in other words, how to generate the training dataset to learn $W_{\text{in}}, W_{\text{out}}$. In this work, we learn these parameters such that similar vector representations can be obtained if the two STL specifications are close to each other in terms of their control policies (for a detailed intuition for this, see Section 4.2). To this end, we generate training dataset by comparing *robustness scores* among the STL specifications, aiming at measuring their closeness. More specifically, for each $\phi_i \in \Phi$, we randomly select an initial state $x_0 \in \mathcal{X}_0$ (following the probability distribution $p : \mathcal{X}_0 \rightarrow \mathbb{R}$), and solve the following maximization problem:

$$\begin{aligned} & \underset{u_0, \dots, u_{T-1}}{\text{maximize}} \quad \rho^{\phi_i}(x_{0:T}) \\ & \text{s.t.} \quad x_{t+1} = f(x_t, u_t), \quad u_t \in \mathcal{U}, \quad t = 0, 1, \dots, T-1. \end{aligned} \quad (4.11)$$

Let $u_0^*, u_1^*, \dots, u_{T-1}^* \in \mathcal{U}$ denote the optimal control inputs as the solution to (4.11), and $x_{0:T}^* = x_0^*, x_1^*, \dots, x_T^*$ the corresponding trajectory of the system (4.1). Thus, the corresponding (maximized) robustness score is $\rho^{\phi_i}(x_{0:T}^*)$. Then, we compute the robustness scores of the obtained trajectory $x_{0:T}^*$

Algorithm 3: Learning $W_{\text{in}}, W_{\text{out}}$ in the skip-gram.

Input : $\Phi = \{\phi_1, \dots, \phi_M\}$ (candidate specifications); x_0 (initial state); P (number of output layers);
 N (dimension of vector representation); N_{ite} (number of iterations)

Output $W_{\text{in}}, W_{\text{out}}$ (weight parameters for the skip-gram)

:

```
1  $\mathcal{D} \leftarrow \emptyset;$ 
2 for each  $\phi_i \in \Phi$  do
3   for  $\ell = 1 : N_{\text{ite}}$  do
4     Select  $x_0 \in \mathcal{X}_0$  according to the probability distribution  $p : \mathcal{X}_0 \rightarrow \mathbb{R};$ 
5     Given  $\phi_i \in \Phi, x_0 \in \mathcal{X}_0$ , solve (4.11) to obtain the optimal trajectory  $x_{0:T}^* = x_0^*, \dots, x_T^*;$ 
6     Compute the robustness scores of  $x_{0:T}^*$  with respect to the other STL specifications (4.12);
7     Sort (4.12) in order of their closeness to  $\rho^{\phi_i}(x_{0:T}^*)$ , and pick up the first  $P$  STL specifications:
        $\phi_{k_1}^i, \phi_{k_2}^i, \dots, \phi_{k_P}^i;$ 
8     Let  $\mathcal{D}_{\text{temp}}$  be given by (4.13). Then, update the dataset as  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{temp}};$ 
9   end
10 end
11 Based on the training data set  $\mathcal{D}$  and the cross entropy loss, train  $W_{\text{in}}, W_{\text{out}}$  via back propagation.
```

with respect to *all the other* STL specifications, i.e.,

$$\rho^{\phi_j}(x_{0:T}^*), \text{ for all } \phi_j \in \Phi \text{ with } i \neq j. \quad (4.12)$$

Then, we sort the robustness scores of (4.12) in order of their closeness to $\rho^{\phi_i}(x_{0:T}^*)$ (i.e., sort by evaluating $|\rho^{\phi_i}(x_{0:T}^*) - \rho^{\phi_j}(x_{0:T}^*)|, i \neq j$). We denote the ordered specifications by $\phi_{k_1}^i, \phi_{k_2}^i, \dots, \phi_{k_{M-1}}^i$, where $\phi_{k_j}^i \in \Phi$ denotes the specification which has j -th closest robustness value to $\rho^{\phi_i}(x_{0:T}^*)$.

Then, we pick up the first P STL specifications, i.e., $\phi_{k_1}^i, \phi_{k_2}^i, \dots, \phi_{k_P}^i$ (recall that P is the number of output layers in the skip-gram model), and set the input and output training data as $\{\phi_i, (\phi_{k_1}^i, \phi_{k_2}^i, \dots, \phi_{k_P}^i)\}$, where ϕ_i represents the input data and $(\phi_{k_1}^i, \phi_{k_2}^i, \dots, \phi_{k_P}^i)$ represents the output data. This data is then encoded by the one-hot vectors, regarded as the training data to learn the skip-gram:

$$\left\{ e_{\phi_i}, \left(e_{\phi_{k_1}^i}, e_{\phi_{k_2}^i}, \dots, e_{\phi_{k_P}^i} \right) \right\}. \quad (4.13)$$

The data in (4.13) is then added to the dataset. For each $\phi_i \in \Phi$, we repeat the above process N_{ite} times so as to obtain a collection of the training data. The proposed approach presented above is summarized in Algorithm 3. Then, the weight parameters $W_{\text{in}}, W_{\text{out}}$ are learned by minimizing the cross entropy loss via back-propagation.

Remark 7. Note that, due to the definition of the robustness given in Section 4.1, the robustness can be nested with max/min functions and thus the optimization problem (4.11) can be non-differentiable. To deal with such a problem, we adopt a smooth approximation of the min/max operators by the log-sum-exp as follows: $\max(a_1, \dots, a_m) \approx \frac{1}{\beta} \ln \sum_{i=1}^m \exp(\beta a_i)$ and $\min(a_1, \dots, a_m) \approx \frac{1}{\beta} \ln \sum_{i=1}^m \exp(-\beta a_i)$, where $\beta > 0$ is the scaling parameter. It is known that the approximation error goes to 0 as $\beta \rightarrow \infty$ (see, e.g., [115]). This approximation allows the problem (4.11) to be solved based on gradient-based algorithms. $\square$

Remark 8. Since the optimization problem (4.11) is non-linear, finding the global optimal solution of the problem (4.11) is difficult and some local optima that leads to undesirable control performance can be obtained. To the best of author's knowledge, there are no general, systematic methodologies to obtain the global optimum of (4.11). However, we believe that there are some heuristic methods that potentially improve the solution. For example, in the case study of Section 4.5, we have confirmed that we could increase the possibility to obtain the reasonable solution (the solution with the positive robustness) by appropriately setting the parameter β in the smooth approximation. Furthermore, we have also confirmed that we could deal with the problem by solving (4.11) several times with the initial states randomly sampled from the vicinity of x_0 , i.e., once x_0 is chosen, (4.11) is solved with the initial state $\tilde{x}_0$ newly sampled satisfying $\|\tilde{x}_0 - x_0\| \leq \epsilon$, where ϵ is a given small positive constant, which could increase the possibility to obtain positive robustness. $\square$

Finally, similarly to Word2vec, the vector representation of the STL specifications is given through the projection of W_{in} , i.e., the vector representation of $\phi_i \in \Phi$ is given by $z_{\phi_i} = W_{\text{in}} e_{\phi_i}$.

Example 1. Consider $M = 5$ candidate STL specifications $\Phi = \{\phi_1, \phi_2, \dots, \phi_5\}$ and $P = 2$. Suppose that we consider ϕ_1 and solve the corresponding optimization problem (4.11) (i.e., $i = 1$) to obtain the optimal trajectory $x_{0:T}^*$. Suppose that the robustness scores of $x_{0:T}^*$ with respect to the STL specifications are obtained as $\rho^{\phi_1}(x_{0:T}^*) = 0.3$, $\rho^{\phi_2}(x_{0:T}^*) = 0.2$, $\rho^{\phi_3}(x_{0:T}^*) = -0.5$, $\rho^{\phi_4}(x_{0:T}^*) = 0.1$, and $\rho^{\phi_5}(x_{0:T}^*) = 0.25$. Then, we select the top $P = 2$ specifications among $\{\phi_2, \phi_3, \phi_4, \phi_5\}$ whose robustness scores are the closest to the one of ϕ_1 . In this case, we select ϕ_5 and ϕ_2 , since the robustness scores of ϕ_5 and ϕ_2 have the most and the second closest robustness values to $\rho^{\phi_1}(x_{0:T}^*)$, respectively. Thus, the resulting data is $\{(\phi_1), (\phi_5, \phi_2)\}$ and the encoded training data to be added in the dataset is $\{[1, 0, 0, 0, 0], ([0, 0, 0, 0, 1], [0, 1, 0, 0, 0])\}$. $\square$

Remark 9. If there exist two or more STL specifications that have the same robustness values in the output data, we add all the combinations to the training data set. For example, in Example 1, if the

robustness values are obtained as $\rho^{\phi_1}(x_{0:T}^*) = 0.3$, $\rho^{\phi_2}(x_{0:T}^*) = 0.2$, $\rho^{\phi_3}(x_{0:T}^*) = -0.5$, $\rho^{\phi_4}(x_{0:T}^*) = 0.2$, and $\rho^{\phi_5}(x_{0:T}^*) = 0.1$, then we add the training data as $\{(\phi_1), (\phi_2, \phi_5)\}$ and $\{(\phi_1), (\phi_4, \phi_5)\}$. $\square$

4.4.2 Training RNN

Here, we describe a detailed procedure of Step (ii). Recall that in Step (ii), we aim to train RNN, whose inputs are the trajectory $x_{0:t}$ and the vector representation of ϕ_i (i.e., z_{ϕ_i}), and the output is the control input u_t . Further, remember that the concrete unfolded structure of the RNN is shown in Fig. 4.1, and the update rules are given by (4.5)–(4.6). Here, the input variable s_t is given by the concatenation of the state x_t and the vector representation of the chosen STL specification, i.e., $s_t = [x_t^\top, z_{\phi_i}^\top]^\top$, where $z_{\phi_i} \in \mathbb{R}^N$ is the vector representation of the chosen STL specification ϕ_i . Now, let $\tilde{\pi}(x_{0:t}, z_{\phi_i}; W_1, W_2)$ denote the control policy (or a mapping) for the RNN, where W_1, W_2 are the RNN parameters to be learned (for the illustration, see Fig. 4.3b). Since the parameters for STL2vec W_{in} is fixed, we here focus on learning the RNN parameters W_1, W_2 to characterize $\tilde{\pi}$.

To numerically solve Problem 1, we keep repeating the following procedure until a certain number of epochs N_{epo} is reached. First, in each epoch, we construct *mini-batch* of the specifications by randomly rearranging the order of the candidate specifications and splitting them up according to the prespecified batch size N_b , i.e., we construct the batches $\mathcal{B}_1 = \{\phi_1^1, \dots, \phi_{N_b}^1\}$, $\mathcal{B}_2 = \{\phi_1^2, \dots, \phi_{N_b}^2\}$, $\dots$, $\mathcal{B}_K = \{\phi_1^K, \dots, \phi_{N_b}^K\}$, where $K = \frac{M}{N_b}$ (assuming that M can be divided by N_b)¹. Then, we iterate the following procedures for all the batches $\mathcal{B}_p$ ($p = 1, \dots, K$) to update the parameters W_1 and W_2 .

Forward Computation

For all the pairs of the initial state x_0^j ($j = 1, \dots, L$) which are randomly sampled from the initial region $\mathcal{X}_0$ and the vectors of the specifications $z_{\phi_i^p}$ ($i = 1, \dots, N_b$) in a batch $\mathcal{B}_p$, we generate the trajectories $x_{0:T}^{i,j}$ by iteratively applying $x_{t+1}^{i,j} = f\left(x_t^{i,j}, \tilde{\pi}\left(x_{0:t}^{i,j}, z_{\phi_i^p}; W_1, W_2\right)\right)$ for fixed W_1 and W_2 from the initial state $x_0^{i,j} = x_0^j$. Then, by using all the $N_b L$ trajectories obtained above, we compute the following loss:

$$-\frac{1}{N_b L} \sum_{i=1}^{N_b} \sum_{j=1}^L \rho^{\phi_i^p}\left(x_{0:T}^{i,j}\right), \quad (4.14)$$

which is an approximation of the expectation (4.3) in Problem 1 with the specifications in a batch (note that we take the negative of the robustness to define a loss).

¹If M is not divisible by N_b , we construct minibatch with $K = \lfloor \frac{M}{N_b} \rfloor$ ($\lfloor \cdot \rfloor$ denotes a floor function) and append the remaining specifications in the last batch. For example, if $\Phi = \{\phi_1, \phi_2, \dots, \phi_5\}$, and $N_b = 2$, we construct minibatch e.g., $\mathcal{B}_1 = \{\phi_1, \phi_3\}$, $\mathcal{B}_2 = \{\phi_2, \phi_4, \phi_5\}$.

Backward Computation

After the forward computation, we compute the gradients for all the parameters via backpropagation through time (BPTT) [120]. This computation can be easily implemented by combining the auto-differentiation tools designed for NNs like PyTorch [141] and STLPG [142] which is a newly developed python toolbox for computing the STL robustness using computation graph.

Weight Update

Lastly, we update all the parameters in the RNN controller by using the weights obtained above. In this study, we use Adam optimizer [143].

4.4.3 Some comparisons between the proposed approach and the approach of learning the RNNs one by one

As one of the alternative methods, one could consider learning a set of RNN controllers independently *one by one* for all the STL specifications. This one-by-one approach might eventually provide higher control performance than the proposed method, since it tries to learn a set of RNN controllers independently for all the candidate STL specifications (on the other hand, the proposed method learns the controller by using only one RNN model). However, the proposed approach has the potential to require much less computational memory than the one-by-one approach due to the following reason. In the proposed method, the total number of the parameters to be learned is given by $N_{\text{in}} + N_{R1} + N_{R2}$, where N_{in} , N_{R1} , and N_{R2} are the number of elements in W_{in} , W_1 , and W_2 . For instance, when we use the Long-Short-Term-Memory (LSTM) as the RNN model, N_1 and N_2 are given by $N_{R1} = 4N_h N_{\text{lstm}}(n + N)$ and $N_{R2} = 4N_h N_{\text{lstm}}m$, where N_h is the dimension of the hidden state of RNN and N_{lstm} is the number of LSTM layers. The number of parameters for STL embedding N_{in} is given by $N_{\text{in}} = MN$. On the other hand, the required number of parameters when we train the controller one-by-one for each candidate specifications is given by $M(N'_1 + N_2)$, where $N'_1 = 4N_h N_{\text{lstm}}n$. From the above, the number of parameters is proportional to the number of STL specifications M for both the proposed approach and the one-by-one approach. However, it is noted that we typically have $N \ll N'_{R1} + N'_{R2}$ ², i.e., the size of the vector representation of STL is selected much smaller than

²It is argued that $N \ll N'_{R1} + N'_{R2}$ indeed holds for many control problems with STL tasks based on the fact that the previous works regarding the control synthesis with RNN (e.g., in [44]) use the structure of 2 LSTM layers with 32 dimensional hidden states, which leads to $N'_{R1} + N'_{R2} = 1280$, even for simple 2-D path planning problem. On the other hand, in the literature of Word2vec [121], vector size N for word embedding is typically set from 50 to 200 for very large corpus like 780×10^3 (in our case study of Section 4.5, we have set $N = 20$ that is smaller than these typical values, since the total number of the candidate STL specifications considered in the case study is $369 \ll 780 \times 10^3$).

the number of the RNN parameters, and thus the number of the required parameters by the proposed approach is smaller than the one-by-one approach. For example, in the case study of Section 4.5, it is shown that the total number of parameters to learn the controller by the proposed approach is 24 times smaller than the one-by-one approach; for details, see Section 4.5.

4.5 Case Study

We show the efficacy of the proposed method through the example of a path planning problem of a vehicle in 2D space. We used PyTorch package [141] for the implementation of RNN, IPOPT in CasADi [129] to solve optimization (4.11), and STL2vec [142] for the computation regarding STL robustness. We used Windows 10 with a 2.80 GHz Core i7 CPU and 32 GB of RAM for all the experiments discussed in this section. Throughout this section, we consider the following non-linear, discrete-time nonholonomic system: $q_{x,t+1} = q_{x,t} + v_t \sin \theta_t$, $q_{y,t+1} = q_{y,t} + v_t \cos \theta_t$, $\theta_{t+1} = \theta_t + \omega_t$, where $q_{x,t}$, $q_{y,t}$ represent position of the vehicle, θ_t is the heading angle, v_t and ω_t are velocity and angular velocity, respectively. The state x_t and control input u_t are defined as $x_t = [q_{x,t}, q_{y,t}, \theta_t]^\top$ and $u_t = [v_t, \omega_t]^\top$, respectively. As illustrated in Fig. 4.4, we consider 4 regions in a 2D space: Reg 1 = $[3, 5] \times [7, 9]$, Reg 2 = $[3, 5] \times [3, 5]$, Reg 3 = $[7, 9] \times [3, 5]$, Reg 4 = $[7, 9] \times [7, 9]$, and the set of initial states $X_0 = [0, 0.7] \times [0, 0.7]$ (blue-edged color region). In addition, we consider 4 sub-regions in each region, which are labeled by indices as shown in Fig. 4.4. In the following, we denote j -th sub-region ($j = 1, 2, 3, 4$) in region i ($i = 1, 2, 3, 4$) as Reg (i, j) .

The STL candidate specifications are shown in Table 4.1. Here, the specifications are considered for all $i, i_1, i_2, i_3 \in \{1, \dots, 4\}$ ($i_1 > i_2$, $i_3 \neq 1$, $i_1 \neq i_3$) and $j, j_1, j_2, j_3 \in \{1, \dots, 4\}$. The total number of the STL candidate specifications is given by 369.

4.5.1 Training STL2vec

Now we train STL2vec based on the proposed approach presented in Section 4.4.1. The parameters for the skip-gram are $N = 20$ and $P = 2$. N_{ite} in Algorithm 3 and the number of epochs for the training of skip-gram model are set to 1 and 100, respectively. After the training, we evaluate similarities between all the 2 different vector representations z_{ϕ_i}, z_{ϕ_j} ($i \neq j$) by the cosine similarity, which is defined as $(z_{\phi_i}^\top z_{\phi_j}) / (\|z_{\phi_i}\| \|z_{\phi_j}\|)$ and it takes the maximum value of 1 if z_{ϕ_i} has the same orientation as z_{ϕ_j} . The time required to train STL2vec is summarized in Table 4.2. In Table 4.3, we illustrate STL specifications that have the largest to fourth-largest cosine similarity values for some example specifications.

We summarize some characteristics that we have observed in the obtained embeddings in the

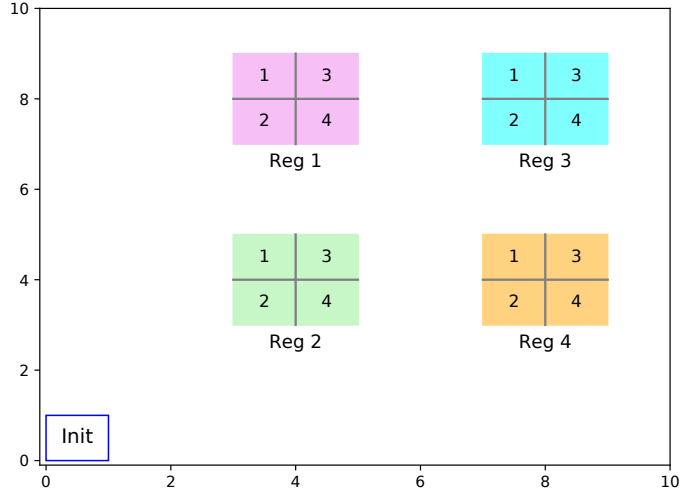


Figure 4.4: Regions and sub-regions in a 2D space.

Table 4.1: STL candidate specifications

Specifications
(a) $F_{[0,20]} \text{Reg } (i, j)$
(b) $F_{[0,20]} \text{Reg } (i_1, j_1) \vee F_{[0,20]} \text{Reg } (i_2, j_2)$
(c) $F_{[0,10]} \text{Reg } (i_1, j_1) \wedge F_{[11,20]} \text{Reg } (i_3, j_2)$
(d) $F_{[0,15]} G_{[0,5]} \text{Reg } (i, j)$
(e) $F_{[0,15]} G_{[0,5]} \text{Reg } (i_1, j_1) \vee F_{[0,15]} G_{[0,5]} \text{Reg } (i_2, j_2)$
(f) $F_{[0,20]} \text{Reg } (4, 4) \wedge (\neg \text{Reg } (4, 4) U_{[0,20]} \text{Reg } (2, 3))$

followings: (I) Each specification in (a) of Table 4.1 is typically embedded close to the corresponding specification in (d). Moreover, each specification in (b) and (e) is embedded close to either $F_{[0,20]} \text{Reg } (i_1, j_1)$ (and $F_{[0,15]} G_{[0,5]} \text{Reg } (i_1, j_1)$) or $F_{[0,20]} \text{Reg } (i_2, j_2)$ (and $F_{[0,15]} G_{[0,5]} \text{Reg } (i_2, j_2)$). Typically, (b) and (e) are embedded close to the specification regarding the region which is closer to init region. For example, as we can partially see in Ex 1 of Table 4.3, we have observed that almost all the specifications $F_{[0,20]} \text{Reg } (2, 2) \vee F_{[0,20]} \text{Reg } (i, j)$ and $F_{[0,15]} G_{[0,5]} \text{Reg } (2, 2) \vee F_{[0,15]} G_{[0,5]} \text{Reg } (i, j)$ ($i \in \{1, 3, 4\}$, $j \in \{1, 2, 3, 4\}$) are embedded close to the specification $F_{[0,20]} \text{Reg } (2, 2)$. This is an intuitive and desirable result when we train the controller since all of these specifications are satisfied by entering whether $\text{Reg } (i_1, j_1)$ or $\text{Reg } (i_2, j_2)$ and stay there more than 5 steps, which is possible actions for all of regions in this example. (II) Each specification in (c) of Table 4.1 is basically embedded close to the specifications which are satisfied by similar trajectory. Specifically, the properties same as the following examples are observed for all

Table 4.2: The time required to obtain STL2vec (in sec)

Procedure	Run-time
Dataset generation (STL2vec)	1452
Training STL2vec	87

Table 4.3: Cosine similarities of vector representations of the STL candidate specifications

Ex 1	$F_{[0,20]} \text{Reg} (2, 2)$	sim		Ex 2	$F_{[0,10]} \text{Reg} (1, 3) \wedge F_{[11,20]} \text{Reg} (3, 1)$	sim
1	$F_{[0,20]} \text{Reg} (3, 3) \vee F_{[0,20]} \text{Reg} (2, 2)$	0.99		1	$F_{[0,10]} \text{Reg} (1, 3) \wedge F_{[11,20]} \text{Reg} (3, 4)$	0.99
2	$F_{[0,20]} \text{Reg} (2, 2) \vee F_{[0,20]} \text{Reg} (1, 1)$	0.99		2	$F_{[0,10]} \text{Reg} (1, 3) \wedge F_{[11,20]} \text{Reg} (3, 3)$	0.99
3	$F_{[0,15]} G_{[0,5]} \text{Reg} (4, 4) \vee F_{[0,15]} G_{[0,5]} \text{Reg} (2, 2)$	0.99		3	$F_{[0,10]} \text{Reg} (1, 3) \wedge F_{[11,20]} \text{Reg} (3, 2)$	0.95
4	$F_{[0,15]} G_{[0,5]} \text{Reg} (2, 2)$	0.99		4	$F_{[0,15]} G_{[0,5]} \text{Reg} (3, 3) \vee F_{[0,15]} G_{[0,5]} \text{Reg} (1, 3)$	0.76
Ex 3	$F_{[0,10]} \text{Reg} (2, 2) \wedge F_{[11,20]} \text{Reg} (3, 1)$	sim		Ex 4	$F_{[0,20]} \text{Reg} (4, 4) \wedge (\neg \text{Reg} (4, 4) U_{[0,20]} \text{Reg} (2, 3))$	sim
1	$F_{[0,20]} \text{Reg} (3, 1)$	0.96		1	$F_{[0,10]} \text{Reg} (2, 3) \wedge F_{[11,20]} \text{Reg} (4, 4)$	0.81
2	$F_{[0,20]} \text{Reg} (3, 1) \vee F_{[0,20]} \text{Reg} (1, 1)$	0.95		2	$F_{[0,10]} \text{Reg} (4, 4) \wedge F_{[11,20]} \text{Reg} (3, 3)$	0.73
3	$F_{[0,20]} \text{Reg} (4, 4) \vee F_{[0,20]} \text{Reg} (3, 1)$	0.95		3	$F_{[0,10]} \text{Reg} (4, 4) \wedge F_{[11,20]} \text{Reg} (3, 1)$	0.71
4	$F_{[0,15]} G_{[0,5]} \text{Reg} (4, 4) \vee F_{[0,15]} G_{[0,5]} \text{Reg} (3, 1)$	0.93		4	$F_{[0,10]} \text{Reg} (4, 4) \wedge F_{[11,20]} \text{Reg} (3, 4)$	0.71

the specifications in (c): (i) specifications $F_{[0,10]} \text{Reg} (1, 3) \wedge F_{[11,20]} \text{Reg} (3, j)$ ($j \in \{1, 2, 3, 4\}$) are embedded close to each other (see Ex 2 of Table 4.3); (ii) as we can see in Ex 3 of Table 4.3, the specifications $F_{[0,10]} \text{Reg} (2, 2) \wedge F_{[11,20]} \text{Reg} (3, 1)$ and $F_{[0,20]} \text{Reg} (3, 1)$ are mapped onto similar vectors (these specifications are satisfied with the same trajectory since $\text{Reg}(2, 2)$ is on the way from the starting point to $\text{Reg}(3, 1)$). (III) The specification (f) of Table 4.1 is embedded close to $F_{[0,10]} \text{Reg} (2, 3) \wedge F_{[11,20]} \text{Reg} (4, 4)$ as we can see from EX 4 in Table 4.3. This is also desirable result since the specifications (f) basically require to firstly reach $\text{Reg} (2, 3)$ and then reach $\text{Reg} (4, 4)$ after that.

4.5.2 Training RNN

Next, we evaluate the control performance of the proposed method. In this experiment, we train the parameters of RNN model for the specifications (a), (b), (c), (f) and $F_{[0,15]} G_{[0,5]} \text{Reg} (1, 1) \vee F_{[0,15]} G_{[0,5]} \text{Reg} (3, 2)$ in (e) in Table 4.1. As for the specifications in (c), we only consider the specifications with $(i_1, i_3) = (1, 3), (2, 1), (2, 3), (2, 4), (4, 3)$. Thus the total number of specifications is 194. We retrain the STL embeddings (20 dimension) with these specifications and use them as the input to the RNN. Same as the previous work [44], all of the RNN models used in this example consists of 2 LSTM layers with 32-dimensional hidden states. As summarized in Table 4.4, we consider the following 3 alternative approaches for comparison with the proposed method: (A1)

Table 4.4: Some alternative approaches

Approaches
(A1) Integer encoding scheme
(A2) One-hot encoding scheme
(A3) Training controllers one-by-one

Table 4.5: Run time needed to reach several average of robustness (in sec)

approach \ average of robustness	0.1	0.15	0.2	0.22
Proposed method	3494	4054	6722	14668
Learning one-by-one	5686	7663	10336	12168
One-hot encoding	27238	33863	—	—
Integer encoding	—	—	—	—

Integer encoding scheme: we incrementally assign an integer number for each STL specification (for example, $\mathbf{F}_{[0,20]}$ Reg (1, 1) is assigned by 1, $\mathbf{F}_{[0,20]}$ Reg (1, 2) is assigned by 2, *etc*), and use it as the input to the RNN (instead of the vectors generated by STL2vec) (A2) One-hot encoding scheme: we generate and assign 194 dimensional one-hot vectors for all the 194 specifications (for example, $\mathbf{F}_{[0,20]}$ Reg (1, 1) is assigned by $[1, 0, \dots, 0]$, $\mathbf{F}_{[0,20]}$ Reg (1, 2) is assigned by $[0, 1, \dots, 0]$, *etc*) (A3) Training one-by-one: we ready 194 RNN models and train each specification one-by-one. When we train the controllers for the approaches (A1) and (A2), we used the same training procedure discussed in Section 4.4.2 and used integer numbers or one-hot vectors instead of the vectors obtained by STL2vec. In both the proposed approach and (A1), (A2), we set the batch size and the number of initial states sampled for training in each iteration (N_b and L in subsection 4.4.2) to 8 and 3, respectively. In (A3), for each epoch we generate initial states x_0^j ($j = 1, \dots, L = 3$) randomly from $\mathcal{X}_0$ and update the RNN parameters assigned for each STL specification $\phi_i \in \Phi$ ($i = 1, \dots, M$) one by one via forward/backward computation (similarly to the procedure of subsection 4.4.2) with the following loss: $-\frac{1}{L} \sum_{j=1}^L \rho^{\phi_i} \left(x_{0:T}^j \right)$.

Fig. 4.5 shows the average of robustness values for all the control schemes and Table 4.5 summarizes the actual time (in sec) required to reach the robustness values 0.1, 0.15, 0.2, 0.22 for all the proposed and alternative methods (the symbol "—" indicates that the corresponding average of the robustness value has not been achieved). Furthermore, the resulting few trajectories obtained by applying the RNN controller trained by the proposed method and approach (A3) are plotted in Fig. 4.6 (both controllers were trained 1100 epochs and the trajectories are plotted for 10 initial states newly sampled

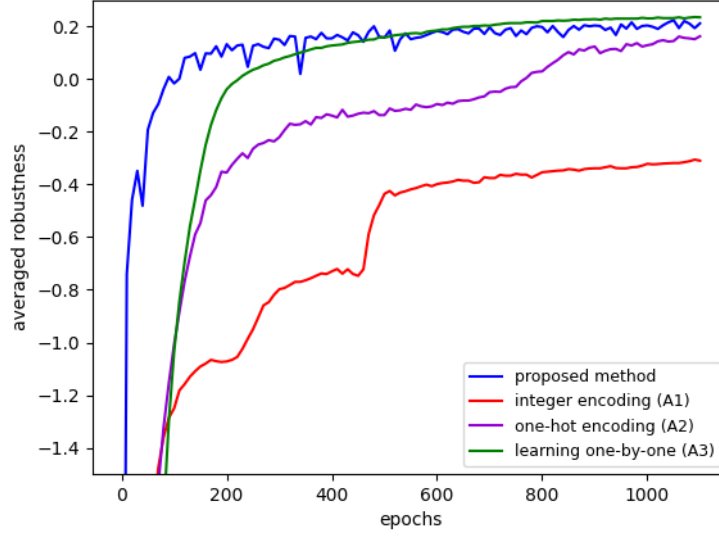
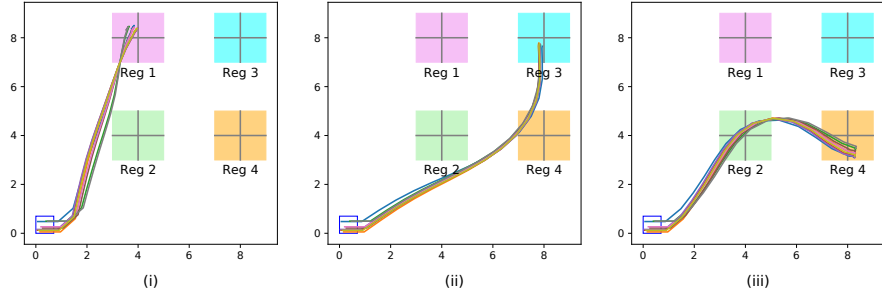
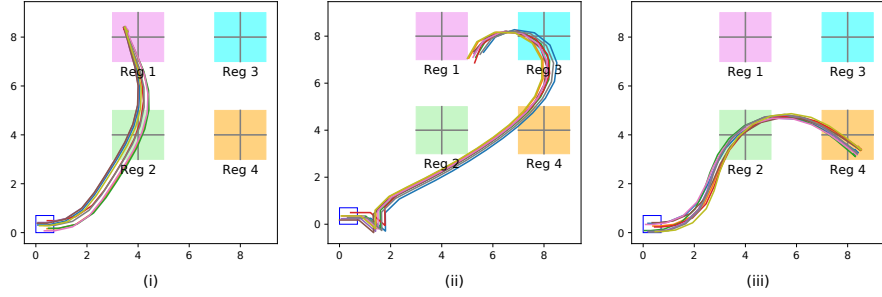


Figure 4.5: The average of robustness values for each control scheme (proposed method and schemes (A1)-(A3)). The average has been taken by testing the controllers for every 10 epochs with 30 initial states randomly sampled from the initial region $\mathcal{X}_0$.

from $\mathcal{X}_0$). The robustness values are collected by testing the controllers once every 10 epochs with 30 initial states newly sampled from the initial region $\mathcal{X}_0$. The values in Fig. 4.5 for the one-by-one scheme (A3) are plotted by taking the mean of the average of the robustness values for all the 194 RNN controllers at the same epochs. We further note here that the actual times required to run one epoch for all the approaches are different from each other. The averaged times required to run the proposed approach and the approaches (A1)-(A3) were 14.2, 14.0, 32.2, and 14.0 [s], respectively (note that the time for approach (A3) is the total sum of the times required to update the parameters of all the RNN models). The time required for the approach (A2) is larger than the other methods because of the large input dimension (the dimension of the one-hot vector is 194). On the other hand, we have confirmed that the time required to update the parameters is not so much increased for the proposed method when $N = 20$ as shown above. As we can see from Fig. 4.5 and Table 4.5, the average of robustness value of the proposed method are improving faster at the beginning of the training procedure than the other methods in terms of both the number of epochs and actual time. Especially, from Table 4.5, we can confirm that the average of robustness value of the proposed method reaches 0.1, 0.15, and 0.2 faster than the other approaches. Moreover, as we mentioned in subsection 4.4.3, we can largely save the memory consumption compared with the scheme (A3). In Table 4.6, we summarize the total number of parameters required for each approach in this example.



(a) Trajectories for the proposed method.



(b) Trajectories for the approach (A3).

Figure 4.6: The testing result for 10 initial states and the 3 specifications: (i) $F_{[0,15]} G_{[0,5]} \text{Reg}(1,1) \vee F_{[0,15]} G_{[0,5]} \text{Reg}(3,2)$, (ii) $F_{[0,10]} \text{Reg}(4,1) \wedge F_{[11,20]} \text{Reg}(3,2)$, (iii) $F_{[0,20]} \text{Reg}(4,4) \wedge (\neg \text{Reg}(4,4) U_{[0,20]} \text{Reg}(2,3))$. Results are shown for both (a) the proposed scheme and (b) approach (A3).

Table 4.6: Total number of parameters in each method

	Number of parameters
Proposed method	10280
approach (A1)	1280
approach (A2)	50432
approach (A3)	248320

However, the average of robustness value of the proposed method is subtly overtaken by the approach (A3) around 500 epoch and the averaged robustness value of the approach (A3) reaches 0.22 faster than the proposed approach. Within 1100 epochs, the maximum average of robustness values of the proposed method and approach (A3) were 0.233 and 0.235, respectively. The reason why the average of robustness values of the proposed method is overtaken by that of the approach (A3) may be because of the effect of the other specifications, i.e., since only one RNN controller is trained for many specifications in the proposed method, the control performance for a specification may be affected by the other specifications, depending on the obtained embedding. Such effect is observed in (b) of Fig. 4.6. The resulting trajectories are converged to the ones that are relatively far from the middle of $\text{Reg}(3, 2)$, which leads to the low robustness although the specification itself is satisfied. To remove such behavior, further investigation for obtaining more superior embedding will be one of our future works.

4.6 Summary

We proposed a method for mapping STL specifications onto the vector space (STL2vec) based on the Word2vec technique. To obtain the STL embeddings that capture the similarities in specifications in terms of control policy, we have provided a method for constructing the dataset by solving the robustness maximization problem for all the candidate specifications. Then, we trained the RNN controller whose inputs are the state trajectory and a vector generated by STL2vec to deal with multiple STL specifications with one RNN model. The example shown in the simulation section shows efficacy of the proposed method in terms of memory consumption and the time required for the training. In this work, it is assumed that the chosen STL specification is fixed during control execution and not allowed to change during the execution. Hence, future work should involve investigating the case where the STL specification is changed during control execution so as to provide more flexibility

of the RNN controller.

Chapter 5

Conclusion

5.1 Summary

This dissertation advances learning-based control for constrained and safety-critical systems by developing controller design methodologies that (i) exploit data efficiently, (ii) explicitly quantify and manage uncertainty, and (iii) preserve formal properties such as feasibility, stability, and safety. Across Chapters 2–4, the unified theme is to bridge modern learning and representation techniques with control-theoretic structures so that learning improves performance and flexibility without compromising constraint satisfaction.

From this perspective, the dissertation provides three complementary building blocks. Chapter 2 develops a data-driven construction of terminal ingredients (terminal set and terminal cost) for iterative MPC using trajectories from past iterations, including a backward-enlargement mechanism to address practical reachability issues. Under suitable assumptions, recursive feasibility and an ISS-type stability property are established, and iteration-wise performance improvement is demonstrated in case studies. Chapter 3 proposes an online CBF learning framework from sensor measurements, where unsafe regions are modeled by GPIS and updated efficiently via Bayesian meta-learning and uncertainty-aware data selection. The learned safety model is integrated into a CBF–CLF–QP controller, and a probabilistic safety guarantee is derived under standard GP confidence assumptions. Chapter 4 introduces STL2vec, an embedding method that maps STL specifications into a vector space so that similarity reflects induced control behavior. Conditioning an RNN controller on STL2vec enables a single policy to address multiple specifications with improved memory and training efficiency compared to per-specification training.

Together, these contributions form a coherent methodology stack in which task requirements can be compactly varied, performance is improved using data, and safety is enforced through certificate-based

constraints with uncertainty quantification.

5.2 Future Work and Outlook

Beyond the chapter-specific extensions, several cross-cutting research directions naturally emerge from this dissertation. In particular, since Chapters 2–4 develop complementary building blocks for data-driven performance improvement, uncertainty-aware safety assurance, and specification-aware generalization, an important next step is to investigate how these components can be combined and scaled toward real-world deployment.

1) Toward an integrated framework (MPC + learned safety + STL-aware objectives). A natural next step is to unify the proposed building blocks into a single closed-loop architecture: STL embeddings can parameterize objectives/constraints in MPC, while learned CBFs act as a safety filter or as constraints within MPC. Establishing end-to-end guarantees (recursive feasibility, stability, and probabilistic safety/specification satisfaction) for such integrated designs is an important direction.

2) Less conservative uncertainty quantification and scalable verification. Reducing conservativeness in confidence sets and certificate bounds remains central. This includes tighter confidence ellipsoids/tubes in Chapter 2, sharper calibration and active data selection for GPIS-based predictions in Chapter 3, and data-efficient dataset generation for STL2vec in Chapter 4. Developing scalable verification schemes (e.g., hybrid robust–probabilistic arguments or sample-efficient validation) will be crucial for higher-dimensional systems.

3) Robustness to non-stationarity and distribution shift. Real applications involve changing dynamics, sensor noise, partial observability, and distribution shift. Extending the theoretical and algorithmic tools across the dissertation to handle such non-stationarity systematically—and clarifying when performance and safety guarantees remain valid—is a key future direction.

4) Real-world deployment and computational efficiency. For practical deployment, improving computational efficiency and demonstrating robustness on hardware platforms are essential. This includes fast solvers and warm-starting for iterative MPC, real-time learning and pruning strategies for online CBF learning, and lightweight policy architectures for specification-conditioned control.

5) Extensions to interactive settings (multi-agent and human-in-the-loop). Many safety-critical tasks are inherently interactive (multi-robot, multi-vehicle, or human-robot). Extending iterative MPC,

online safety learning, and specification embeddings to interactive settings with coupled constraints and shared safety models is a promising direction.

6) Theory unification: joint performance–safety–specification guarantees. Finally, developing a unified theoretical lens that connects performance improvement (e.g., regret/suboptimality bounds), constraint satisfaction (recursive feasibility), and safety/specification satisfaction (probabilistic CBF and temporal-logic guarantees) would strengthen the long-term impact of this dissertation and guide principled design choices across tasks and domains.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to Professor Shigemasa Takai, Division of Electrical, Electronic, and Infocommunications Engineering, Graduate School of Engineering, Osaka University, for his dedicated guidance and continuous support throughout my doctoral studies.

I would also like to express my special appreciation to Associate Professor Kazumune Hashimoto, Division of Electrical, Electronic, and Information Engineering, Graduate School of Engineering, Osaka University, for his supervision, for providing me with a wide range of valuable research experiences, and for his discussions and constructive comments, which have been essential to the completion of this thesis.

I am deeply grateful to Associate Professor Naoki Hayashi, Division of Mathematical Social System Science, Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, who provided dedicated supervision and warm encouragement during my master's program, which greatly influenced my subsequent research direction.

I am also grateful to Associate Professor Xun Shen, Graduate School of Engineering, Tokyo University of Agriculture and Technology, and Dr. Akifumi Wachi, LY Corporation, for their valuable advice and fruitful discussions that have inspired and refined my research. In addition, I would like to express my sincere gratitude to Associate Professor Masako Kishida, National Institute of Informatics, for her insightful comments on my papers and for providing me with the opportunity to work as a research assistant under her supervision.

I would also like to express my sincere gratitude to Professor Tsuyoshi Funaki and Professor Tomoo Ushio, Division of Electrical, Electronic, and Infocommunications Engineering, Graduate School of Engineering, Osaka University, for serving as referees for my doctoral thesis and for their valuable comments and suggestions, which have greatly improved this dissertation.

Finally, I would like to thank all members of the Takai laboratory for their support, and my family for their constant encouragement throughout my doctoral studies.

Publication List

Peer-Reviewed Journal Article (First Author)

- [1] W. Hashimoto, N. Hayashi, N. Hara, and S. Takai: Distributed Blade Pitch Angle Control in Floating Offshore Wind Farm with Coordination of Total Power Generation, Transactions of the SICE, vol. 56, no. 8, pp. 395–402, 2020 (in Japanese).
- [2] W. Hashimoto, K. Hashimoto, and S. Takai: STL2vec: Signal Temporal Logic Embeddings for Control Synthesis With Recurrent Neural Networks, IEEE Robotics & Automation Letters, vol. 7, no. 2, pp. 5246–5253, 2022.
- [3] W. Hashimoto, K. Hashimoto, A. Wachi, X. Shen, M. Kishida, and S. Takai: Data-efficient safe learning and control with on-board sensors: Bayesian meta-learning and barrier function based approach, Advanced Robotics, vol. 38, no. 21 pp. 1501–1514, 2024.
- [4] W. Hashimoto, K. Hashimoto, M. Kishida, and S. Takai: Robust learning-based iterative model predictive control for unknown non-linear systems; IET Control Theory & Applications, vol. 18, no. 18, 2540–2554, 2024.
- [5] W. Hashimoto, N. Hayashi, N. Hara, N. Kawaguchi, T. Sato, and S. Takai, Consensus-Based Primal-Dual Method with Event-Triggered Communication, International Journal of Control, doi: <https://doi.org/10.1080/00207179.2025.2537855>, 2025.

Reviewed International Conference Proceeding (First Author)

- [1] W. Hashimoto, K. Hashimoto, Y. Onoue, S. Takai: Learning-Based Iterative Optimal Control for Unknown Systems Using Gaussian Process Regression, European Control Conference (ECC), 2022.

Peer-Reviewed Journal Article (Co-author)

- [1] W. Zhijun, K. Hashimoto, W. Hashimoto, S. Takai: A Lifting Approach to Learning-Based Self-Triggered Control with Gaussian Processes, *Advanced Robotics*, vol. 38, no. 6 pp. 410—420, 2024.

Reviewed International Conference Proceeding (Co-author)

- [1] A. Wachi, W. Hashimoto, X. Shen, K. Hashimoto: Safe Exploration in Reinforcement Learning: A Generalized Formulation and Algorithms, *Neural Information Processing Systems (NeurIPS)*, 2023.
- [2] A Wachi, W Hashimoto, K Hashimoto: Long-term Safe Reinforcement Learning with Binary Feedback, *Association for the Advancement of Artificial Intelligence (AAAI)*, 2024.
- [3] X. Shen, A. Wachi, W. Hashimoto, K. Hashimoto, S. Takai: Safe Reinforcement Learning Using Model Predictive Control with Probabilistic Control Barrier Function, *American Control Conference (ACC)*, 2024.

Bibliography

- [1] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010.
- [2] B. Anderson and J. Moore, *Optimal Control: Linear Quadratic Methods, Augmented Edition*. Dover Publications, 2nd ed., 2007.
- [3] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 3rd ed., 2002.
- [4] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Prentice Hall, 1996.
- [5] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Wiley, 2nd ed., 2005.
- [6] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. Volume 3, 2020, pp. 269–296, 2020.
- [7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2 ed., 2018.
- [8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [10] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv*, vol. 1912.06680, 2019.

- [11] J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 16, pp. 1437–1480, 2015.
- [12] L. Brunke, M. A. Abdolmaleki, Z. Hu, *et al.*, “Safe learning in robotics: From learning-based control to safe reinforcement learning,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.
- [13] M. Forgione and D. Piga, “Continuous-time system identification with neural networks: Model structures and fitting criteria,” *European Journal of Control*, vol. 59, pp. 69–81, 2021.
- [14] M. Lutter, J. Watson, and J. Peters, “Differentiable physics models for real-world offline model-based reinforcement learning,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6166–6173, 2021.
- [15] Y. Qiao, J. Liang, V. Koltun, and M. C. Lin, “Scalable differentiable physics for learning and control,” in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 7847–7856, 2020.
- [16] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, “Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4661–4666, 2016.
- [17] F. Castañeda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, “Pointwise feasibility of gaussian process-based safety-critical control under model uncertainty,” in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6762–6769, 2021.
- [18] L. Hewing, J. Kabzan, and M. N. Zeilinger, “Cautious model predictive control using gaussian process regression,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020.
- [19] K. J. Åström and B. Wittenmark, “On self-tuning regulators,” *Automatica*, vol. 9, no. 2, pp. 185–199, 1973.
- [20] I. D. Landau, *Adaptive Control: The Model Reference Approach*. New York: Marcel Dekker, 1979.
- [21] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, “Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1557–1564, 2017.

- [22] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, “Safe exploration for active learning with gaussian processes,” in *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2015)*, vol. 9286 of *Lecture Notes in Computer Science*, pp. 133–149, Springer, 2015.
- [23] F. Berkenkamp, A. P. Schoellig, and A. Krause, “Safe controller optimization for quadrotors with gaussian processes,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 491–496, 2016.
- [24] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, “Bayesian gait optimization for bipedal locomotion,” in *Learning and Intelligent Optimization (LION 8)*, vol. 8426 of *Lecture Notes in Computer Science*, pp. 274–290, Springer, 2014.
- [25] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, “Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics,” *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.
- [26] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, 2019.
- [27] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, “Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview,” *Annual Reviews in Control*, vol. 52, pp. 135–169, 2021.
- [28] Z. Artstein, “Stabilization with relaxed controls,” *Nonlinear Analysis: Theory, Methods Applications*, vol. 7, no. 11, pp. 1163–1173, 1983.
- [29] E. D. Sontag, “A “universal” construction of artstein’s theorem on nonlinear stabilization,” *Systems & Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.
- [30] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [31] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, 2019.

- [32] S. M. Richards, F. Berkenkamp, and A. Krause, “The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems,” in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, vol. 87 of *Proceedings of Machine Learning Research*, pp. 466–476, PMLR, 2018.
- [33] Y. Chang, N. Roohi, and S. Gao, “Neural lyapunov control,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.
- [34] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, “Synthesis of control barrier functions using a supervised machine learning approach,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7139–7145, 2020.
- [35] S. Liu, C. Liu, and J. Dolan, “Safe control under input limits with neural control barrier functions,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, vol. 205 of *Proceedings of Machine Learning Research*, pp. 1970–1980, PMLR, 2023.
- [36] A. Pnueli, “The temporal logic of programs,” in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 46–57, 1977.
- [37] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems* (Y. Lakhnech and S. Yovine, eds.), (Berlin, Heidelberg), pp. 152–166, Springer Berlin Heidelberg, 2004.
- [38] X. Yin, B. Gao, and X. Yu, “Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges,” *Annual Reviews in Control*, vol. 57, p. 100940, 2024.
- [39] G. E. Fainekos and G. J. Pappas, “Robustness of temporal logic specifications for continuous-time signals,” *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.
- [40] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, “Model predictive control with signal temporal logic specifications,” in *53rd IEEE Conference on Decision and Control*, pp. 81–87, 2014.
- [41] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Cham: Springer, 2017.
- [42] I. Haghghi, N. Mehdipour, E. Bartocci, and C. Belta, “Control from signal temporal logic specifications with smooth cumulative quantitative semantics,” *arXiv preprint*, vol. arXiv:1904.11611, 2019.

- [43] W. Liu, N. Mehdipour, and C. Belta, "Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints," *IEEE Control Systems Letters*, vol. 6, pp. 91–96, 2022.
- [44] W. Liu, M. Nishioka, and C. Belta, "Safe model-based control from signal temporal logic specifications using recurrent neural networks," 2023.
- [45] S. YAGHOUBI and G. FAINEKOS, "Worst-case satisfaction of stl specifications using feed-forward neural network controllers: A lagrange multipliers approach," in *2020 Information Theory and Applications Workshop (ITA)*, pp. 1–20, 2020.
- [46] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
- [47] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.
- [48] J. H. Lee and K. S. Lee, "Iterative learning control applied to batch processes: An overview," *Control Engineering Practice*, vol. 15, no. 10, pp. 1306–1318, 2007. Special Issue - International Symposium on Advanced Control of Chemical Processes (ADCHEM).
- [49] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2018.
- [50] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks: A computationally efficient approach for linear system," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3142–3147, 2017. 20th IFAC World Congress.
- [51] C. E. Rasmussen, *Gaussian Processes in Machine Learning*, pp. 63–71. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
- [52] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.
- [53] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based nmpe enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

- [54] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, “Learning-based model predictive control for autonomous racing,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.
- [55] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, “Learning-based model predictive control for safe exploration,” in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066, 2018.
- [56] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer, “Learning-based robust model predictive control with state-dependent uncertainty,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 442–447, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.
- [57] M. Maiworm, D. Limon, and R. Findeisen, “Online learning-based model predictive control with gaussian process models and stability guarantees,” *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8785–8812, 2021.
- [58] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. V. Dimarogonas, “Learning-based symbolic abstractions for nonlinear control systems,” *Automatica*, vol. 146, p. 110646, 2022.
- [59] E. T. Maddalena, P. Scharnhorst, Y. Jiang, and C. N. Jones, “KPC: Learning-based model predictive control with deterministic guarantees,” in *Proceedings of the 3rd Conference on Learning for Dynamics and Control* (A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A. nbsp;Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, eds.), vol. 144 of *Proceedings of Machine Learning Research*, pp. 1015–1026, PMLR, 07 – 08 June 2021.
- [60] J. Köhler, P. Kötting, R. Soloperto, F. Allgöwer, and M. A. Müller, “A robust adaptive model predictive control framework for nonlinear uncertain systems,” *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8725–8749, 2021.
- [61] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari, “Adaptive receding horizon control for constrained mimo systems,” *Automatica*, vol. 50, no. 12, pp. 3019–3029, 2014.
- [62] M. Lorenzen, M. Cannon, and F. Allgöwer, “Robust mpc with recursive model update,” *Automatica*, vol. 103, pp. 461–471, 2019.
- [63] D. Fan, A. Agha, and E. Theodorou, “Deep learning tubes for tube mpc,” 07 2020.
- [64] Z. Yan and J. Wang, “Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 457–469, 2014.

- [65] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, “Provably safe and robust learning-based model predictive control,” *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.
- [66] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, “Automatic lqr tuning based on gaussian process global optimization,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, p. 270–277, IEEE Press, 2016.
- [67] M. Menner and M. N. Zeilinger, “Convex formulations and algebraic solutions for linear quadratic inverse optimal control problems,” in *2018 European Control Conference (ECC)*, pp. 2107–2112, 2018.
- [68] M. Zanon and S. Gros, “Safe reinforcement learning using robust mpc,” *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2021.
- [69] U. Rosolia, X. Zhang, and F. Borrelli, “Robust learning model predictive control for iterative tasks: Learning from experience,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1157–1162, 2017.
- [70] U. Rosolia, X. Zhang, and F. Borrelli, “Robust learning model-predictive control for linear systems performing iterative tasks,” *IEEE Transactions on Automatic Control*, vol. 67, no. 2, pp. 856–869, 2022.
- [71] B. Thananjeyan, A. Balakrishna, U. Rosolia, J. E. Gonzalez, A. Ames, and K. Goldberg, “Abc-lmpc: Safe sample-based learning mpc for stochastic nonlinear dynamical systems with adjustable boundary conditions,” in *Algorithmic Foundations of Robotics XIV* (S. M. LaValle, M. Lin, T. Ojala, D. Shell, and J. Yu, eds.), (Cham), pp. 1–17, Springer International Publishing, 2021.
- [72] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg, “Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3612–3619, 2020.
- [73] Y. R. Stürz, E. L. Zhu, U. Rosolia, K. H. Johansson, and F. Borrelli, “Distributed learning model predictive control for linear systems,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 4366–4373, 2020.
- [74] W. Hashimoto, K. Hashimoto, M. Kishida, and S. Takai, “Reference-free iterative learning model predictive control with neural certificates,” 2025.

- [75] W. Hashimoto and K. Hashimoto, “Mm-lmpc: Multi-modal learning model predictive control via bandit-based mode selection,” 2025.
- [76] U. Rosolia and F. Borrelli, “Learning how to autonomously race a car: A predictive control approach,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2020.
- [77] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, “Adaptive mpc for iterative tasks,” in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6322–6327, 2018.
- [78] D. Mayne, M. Seron, and S. Raković, “Robust model predictive control of constrained linear systems with bounded disturbances,” *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.
- [79] A. Clark, “Control barrier functions for complete and incomplete information stochastic systems,” in *2019 American Control Conference (ACC)*, pp. 2928–2935, 2019.
- [80] K. Garg and D. Panagou, “Robust control barrier and control lyapunov functions with fixed-time convergence guarantees,” in *2021 American Control Conference (ACC)*, pp. 2292–2297, 2021.
- [81] Y. Chen, H. Peng, and J. Grizzle, “Obstacle avoidance for low-speed autonomous vehicles with barrier function,” *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, 2018.
- [82] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [83] M. A. Khan, T. Ibuki, and A. Chatterjee, “Gaussian control barrier functions: Non-parametric paradigm to safety,” *IEEE Access*, vol. 10, pp. 99823–99836, 2022.
- [84] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, “Synthesis of control barrier functions using a supervised machine learning approach,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7139–7145, 2020.
- [85] K. Long, C. Qian, J. Cortés, and N. Atanasov, “Learning barrier functions with memory for robust safe navigation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.
- [86] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proceedings of the 34th International Conference on Machine Learning, ICML*

2017, Sydney, NSW, Australia, 6-11 August 2017 (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 2017.

- [87] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, “Meta-learning in neural networks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022.
- [88] J. Harrison, A. Sharma, and M. Pavone, “Meta-learning priors for efficient online bayesian regression,” in *Algorithmic Foundations of Robotics XIII* (M. Morales, L. Tapia, G. Sánchez-Ante, and S. Hutchinson, eds.), (Cham), pp. 318–337, Springer International Publishing, 2020.
- [89] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, “Safe active dynamics learning and control: A sequential exploration-exploitation framework,” *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2888–2907, 2022.
- [90] O. Williams and A. Fitzgibbon, “Gaussian process implicit surfaces,” in *Gaussian Processes In Practice*, June 2006.
- [91] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, “Geometric priors for gaussian process implicit surfaces,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 373–380, 2017.
- [92] M. P. Gerardo-Castro, T. Peynot, and F. Ramos, *Laser-Radar Data Fusion with Gaussian Process Implicit Surfaces*, pp. 289–302. Cham: Springer International Publishing, 2015.
- [93] C. Dawson, S. Gao, and C. Fan, “Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control,” *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, 2023.
- [94] A. Taylor, A. Singletary, Y. Yue, and A. Ames, “Learning for safety-critical control with control barrier functions,” in *Proceedings of the 2nd Conference on Learning for Dynamics and Control* (A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, eds.), vol. 120 of *Proceedings of Machine Learning Research*, pp. 708–717, PMLR, 10–11 Jun 2020.
- [95] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath, “Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions,” in *Proceedings of Robotics: Science and Systems*, (Corvalis, Oregon, USA), July 2020.

- [96] P. Jagtap, G. J. Pappas, and M. Zamani, “Control barrier functions for unknown nonlinear systems using gaussian processes,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3699–3704, 2020.
- [97] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, “Control barriers in bayesian learning of system dynamics,” *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 214–229, 2023.
- [98] F. Castañeda, J. J. Choi, W. Jung, B. Zhang, C. J. Tomlin, and K. Sreenath, “Recursively feasible probabilistic safe online learning with control barrier functions,” 2024.
- [99] K. Long, V. Dhiman, M. Leok, J. Cortés, and N. Atanasov, “Safe control synthesis with uncertain dynamics and constraints,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7295–7302, 2022.
- [100] R. K. Cosner, Y. Yue, and A. D. Ames, “End-to-end imitation learning with safety guarantees using control barrier functions,” in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 5316–5322, 2022.
- [101] S. Yaghoubi, G. Fainekos, and S. Sankaranarayanan, “Training neural network controllers using control barrier functions in the presence of disturbances,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2020.
- [102] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, vol. 33, pp. 3387–3395, 2019.
- [103] Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, and M. Egerstedt, “Safe reinforcement learning using robust control barrier functions,” *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2886–2893, 2025.
- [104] Z. Marvi and B. Kiumarsi, “Safe off-policy reinforcement learning using barrier functions,” in *2020 American Control Conference (ACC)*, pp. 2176–2181, 2020.
- [105] W. Jin, Z. Wang, Z. Yang, and S. Mou, “Neural certificates for safe control policies,” 2020.
- [106] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, “Learning control barrier functions from expert demonstrations,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3717–3724, 2020.

- [107] C. Dawson, Z. Qin, S. Gao, and C. Fan, “Safe nonlinear control using robust neural lyapunov-barrier functions,” in *Proceedings of the 5th Conference on Robot Learning* (A. Faust, D. Hsu, and G. Neumann, eds.), vol. 164 of *Proceedings of Machine Learning Research*, pp. 1724–1735, PMLR, 08–11 Nov 2022.
- [108] F. B. Mathiesen, S. C. Calvert, and L. Laurenti, “Safety certification for stochastic systems via neural barrier functions,” *IEEE Control Systems Letters*, vol. 7, pp. 973–978, 2023.
- [109] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning,” 2019.
- [110] S. M. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, “Control-oriented meta-learning,” *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 777–797, 2023.
- [111] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340, PMLR, 09–15 Jun 2019.
- [112] R. Koymans, “Specifying real - time properties with metric temporal logic,” *Real-Time Systems*, vol. 2, no. 3, pp. 255–299, 1990.
- [113] S. Sadraddini and C. Belta, “Robust temporal logic model predictive control,” in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 772–779, 2015.
- [114] N. Mehr, D. Sadigh, R. Horowitz, S. S. Sastry, and S. A. Seshia, “Stochastic predictive freeway ramp metering from signal temporal logic specifications,” in *2017 American Control Conference (ACC)*, pp. 4884–4889, 2017.
- [115] Y. V. Pant, H. Abbas, and R. Mangharam, “Smooth operator: Control using the smooth robustness of temporal logic,” in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1235–1240, 2017.
- [116] X. Li, Y. Ma, and C. Belta, “A policy search method for temporal logic specified reinforcement learning tasks,” in *2018 Annual American Control Conference (ACC)*, pp. 240–245, 2018.
- [117] N. Mehdipour, C.-I. Vasile, and C. Belta, “Arithmetic-geometric mean robustness for control from signal temporal logic specifications,” in *2019 American Control Conference (ACC)*, pp. 1690–1695, 2019.

- [118] I. Haghghi, N. Mehdipour, E. Bartocci, and C. Belta, “Control from signal temporal logic specifications with smooth cumulative quantitative semantics,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4361–4366, 2019.
- [119] A. Balakrishnan and J. V. Deshmukh, “Structured reward shaping using signal temporal logic specifications,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3481–3486, 2019.
- [120] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [121] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in Neural Information Processing Systems*, vol. 26, 10 2013.
- [122] X. Rong, “word2vec parameter learning explained,” 2016.
- [123] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, 2019.
- [124] W. Hashimoto, K. Hashimoto, M. Kishida, and S. Takai, “Robust learning-based iterative model predictive control for unknown non-linear systems,” *IET Control Theory & Applications*, vol. 18, no. 18, pp. 2540–2554, 2024.
- [125] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, “Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4661–4666, 2016.
- [126] I. Steinwart and A. Christmann, *Support Vector Machines*. Information Science and Statistics, Springer New York, NY, 1 ed., 2008.
- [127] A. B. Kurzhanski, “Ellipsoidal calculus for estimation and feedback control,” in *Systems and Control in the Twenty-First Century* (C. I. Byrnes, B. N. Datta, C. F. Martin, and D. S. Gilliam, eds.), (Boston, MA), pp. 229–243, Birkhäuser Boston, 1997.
- [128] L. Grüne and C. M. Kellett, “Iss-lyapunov functions for discontinuous discrete-time systems,” *IEEE Transactions on Automatic Control*, vol. 59, no. 11, pp. 3098–3103, 2014.
- [129] “CasADi.” <https://web.casadi.org/>.

- [130] G. authors (SheffieldML), “GPpy: A gaussian process framework in python.” <https://gpy.readthedocs.io/en/deploy/>, 2024. Accessed: 2025-10-26.
- [131] “Learning-based model predictive control for safe exploration.” <https://github.com/befelix/safe-exploration>. accessed 2025-11-16.
- [132] W. Hashimoto, K. Hashimoto, A. Wachi, X. Shen, M. Kishida, and S. Takai, “Data-efficient safe learning and control with on-board sensors: Bayesian meta-learning and barrier function based approach,” *Advanced Robotics*, vol. 38, no. 21, pp. 1501–1514, 2024.
- [133] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, p. 359–366, jul 1989.
- [134] Y. Abbasi-yadkori, D. Pál, and C. Szepesvári, “Improved algorithms for linear stochastic bandits,” in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.
- [135] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, “Faithful euclidean distance field from log-gaussian process implicit surfaces,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2461–2468, 2021.
- [136] D. Nguyen-Tuong and J. Peters, “Incremental online sparsification for model learning in real-time robot control,” *Neurocomputing*, vol. 74, no. 11, pp. 1859–1867, 2011. Adaptive Incremental Learning in Neural Networks Learning Algorithm and Mathematic Modelling Selected papers from the International Conference on Neural Information Processing 2009 (ICONIP 2009).
- [137] L. Brunke, S. Zhou, M. Che, and A. P. Schoellig, “Optimized control invariance conditions for uncertain input-constrained nonlinear control systems,” *IEEE Control Systems Letters*, vol. 8, pp. 157–162, 2024.
- [138] “PyBullet.” <https://pybullet.org/wordpress/>.
- [139] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. V. Dimarogonas, “Learning-based symbolic abstractions for nonlinear control systems,” *Automatica*, vol. 146, p. 110646, 2022.
- [140] W. Hashimoto, K. Hashimoto, and S. Takai, “Stl2vec: Signal temporal logic embeddings for control synthesis with recurrent neural networks,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5246–5253, 2022.

- [141] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS 2017 Workshop on Autodiff*, 2017.
- [142] K. Leung, N. Aréchiga, and M. Pavone, “Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods,” *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 356–370, 2023.
- [143] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.