Enhancing Safety and Performance in Multi-agent Systems and Soft Robots via Reinforcement Learning

Yogita Choudhary

CMU-RI-TR-25-70

July 22, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. John Dolan, *Chair*
Prof. Guanya Shi, *Chair*
Prof. Changliu Liu
Simin Liu

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my parents and my sister.

Abstract

With the growing deployment of robots across various safety-critical applications, ensuring safety has become increasingly essential to prevent accidents and enhance reliability. Addressing safety in robotic systems presents several inherent challenges.

Firstly, in multi-agent settings, incorporating safety filters can adversely impact overall task performance, as agents must simultaneously ensure collision avoidance and timely task completion. To mitigate this trade-off, we propose a co-safe Reinforcement Learning (RL) framework wherein the ego agent utilizes the neighbouring agents' safety model information to better inform its policy, thereby minimally influencing other agents' trajectories, resulting in improved task performance on the defined metrics.

Secondly, ensuring safety in soft robots is particularly challenging due to their inherently complex and highly nonlinear dynamics, which limit the effectiveness of traditional model-based safety filters. To address this, a model-free safety filter based on Q-learning is introduced, designed to integrate seamlessly with standard reinforcement learning frameworks. The practical viability and robustness of the proposed safety filter is demonstrated through simulation studies and real-world validations using a soft robotic limb actuated by shape memory alloys.

Acknowledgments

First and foremost, I would like to thank my advisors, Prof. John M. Dolan and Prof. Guanya Shi, for their invaluable guidance and continuous support. I also sincerely thank Prof. Changliu Liu for her encouragement and valuable feedback that helped shape the direction of my work.

I'm grateful to everyone in the LeCAR Lab and the DRIVE Lab for their constant assistance and encouragement. Special thanks go to my collaborators—Andrew Sue, Guanqi He, Maitham F. Al-Sunni, Yiwei Lyu, and Richard Desatnik. I have learned a lot from each one of them.

I'd like to extend my gratitude to my labmates for their support, camaraderie, and insightful discussions. Listed here alphabetically: André Gomes, Anoushka Alavilli, Chaitanya Chawla, Chaoyi Pan, Giri Anantharaman, Haotian Lin, Haoyang Weng, Juan Alvarez-Padilla, Kai S. Yun, Nikhil Sobanbabu, Simin Liu, Tairan He, Wenli Xiao, Xiaofeng Guo, Yitang Li, Yuanhang Zhang, Zeji Yi.

Finally, my deepest thanks go to my parents and my sister, whose unwavering love, encouragement, and support have made this journey possible. I am deeply grateful for the countless sacrifices they have made to support my dreams.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing deployment of robots in various safety-critical applications, ranging from autonomous transportation [53] to medical procedures [25], underscores the vital need for robust safety mechanisms. As robotic systems evolve in complexity and autonomy, ensuring safety without compromising performance becomes an intricate challenge. This thesis explores novel methodologies aimed at enhancing both safety and performance in robotic systems, specifically addressing two distinct domains: multi-agent systems and soft robots.

Primary safety-enforcement methods typically employ classical control-theoretic approaches such as Control Barrier Functions (CBFs) [5, 31], Hamilton-Jacobi reachability analysis [7, 28, 29], Lyapunov-based methods [35], and Model Predictive Control (MPC) [23, 37]. These methods utilize a minimally restrictive supervisory control strategy and generate a reactive fallback policy that steps in as a final safeguard, supplanting the primary controller only when a safety-critical situation (e.g., an imminent collision) is detected.

While such real-time interventions are essential, they often induce overly conservative behaviors in multi-agent settings, causing agents to frequently deviate from their intended paths. This can lead to increased task durations, degraded formation-keeping, and diminished collective performance. To address this inherent trade-off, we introduce a co-safe reinforcement learning framework in Chapter 2 in which an ego agent leverages neighbouring agents' safety model information to better inform its policy for enhancing overall task performance. By observing each neighbour's CBF decay rate and safety model, the

ego agent influences other agents' trajectories courteously and enhances collective task performance.

Although classical control-theoretic tools for ensuring safety, such as those previously described, provide rigorous guarantees, they typically require accurate dynamical models for providing safety guarantees. However, obtaining precise models can be challenging or infeasible for highly uncertain and complex systems like soft robots. To overcome these limitations, we introduce a model-free safety filter in Chapter 3, grounded in Q-learning. Our proposed approach maintains theoretical safety guarantees under optimal conditions and demonstrates robustness in suboptimal scenarios, integrating seamlessly into standard reinforcement learning pipelines. The robustness and effectiveness of our proposed method are validated through extensive simulations and real-world hardware experiments involving a soft robotic limb actuated by shape memory alloys.

Chapter 2

Co-safety in Multi-agent Systems

2.1 Introduction

Multi-agent system control design can broadly be classified into two categories: **centralized** and **decentralized** control [49]. Centralized policies rely on the global state of the multi-agent system to dictate the decisions of all agents [10], whereas decentralized policies operate using only local or partial knowledge of the system [21, 22, 52].

Within decentralized control, two common subcases arise: 1) all agents are controllable; and 2) only the "ego" agent is controllable, with other agents treated as dynamic obstacles. Since most real-world scenarios such as autonomous driving, drone navigation, and crowd interaction fall into the second category, this work focuses primarily on the ego-agent control setting.

Ensuring collision-free navigation in such scenarios is challenging because purely reactive safety mechanisms, such as Control Barrier Function (CBF) filters or Hamilton-Jacobi reachability approaches, often induce overly conservative behaviour. Agents may deviate significantly from their nominal paths, leading to longer task completion times and degraded collective performance.

To address this trade-off between safety and efficiency, we propose a co-safe reinforcement learning (RL) framework. In this approach, the ego agent explicitly leverages neighbouring agents' CBF information to inform its policy. By anticipating how its actions influence nearby agents, the ego agent exhibits courteous, co-safe behaviour - maintaining safety while minimizing disruption to others' trajectories. This strategy leads to smoother,

more efficient multi-agent coordination and shorter overall task completion times.

In this chapter, we present the motivation, methodology, and experimental evaluation of our co-safe RL framework:

1. We introduce reward shaping to jointly encourage goal-reaching, safe as well as co-safe behaviour.
2. We demonstrate through simulation experiments that leveraging CBF information significantly improves trajectory efficiency and task completion time compared to conventional approaches.

2.2 Related Work

Several prominent approaches have been developed to address the multi-agent collision avoidance problem. The Velocity Obstacle (VO) method, a widely adopted strategy, defines collision cones in velocity space representing potential collisions within a future time horizon. This method assumes that agents move piecewise linearly and obstacles maintain constant velocities over the considered horizon [14]. Buffered Voronoi Cells (BVC), another prevalent approach, provides formal safety guarantees using only positional information from neighbouring agents. However, BVC methods are typically constrained to systems characterized by linear dynamics, limiting their applicability to more complex robotic platforms [36, 54].

Alternatively, Hamilton-Jacobi-Isaacs (HJI) reachability analysis has been employed to ensure collision avoidance rigorously by solving HJI partial differential equations, though these methods often incur significant computational costs [7, 11]. Control Barrier Functions (CBFs) have emerged as a powerful framework, offering formal safety guarantees by imposing point-wise linear constraints on control inputs. These constraints enforce forward invariance of a predefined safe set, ensuring that agents remain collision-free throughout operation [4, 45]. Utilizing CBFs and reachability analysis for ensuring safety results in overly conservative behaviours as these methods utilize a minimally restrictive supervisory control strategy and generate a reactive fallback policy that steps in as a final safeguard, supplanting the primary controller only when a safety-critical situation (e.g., an imminent collision) is detected.

Recent advancements in the multi-agent community have focused on leveraging neigh-

bouring agents' state information to improve task performance, while still maintaining safety via CBF filters. For example, [17, 26] propose online adaptation of CBF parameters utilizing neighbouring agents' states to mitigate deadlocks effectively. Furthermore, [33] enhances task efficiency by dynamically adjusting the CBF decay rate—the rate at which agents approach the safety boundary—based on a trust metric reflecting other agents' cooperativeness.

Similarly, [27] proposes a framework to dynamically adapt the decay rate of the ego agent by leveraging safety model information from neighbouring agents. Their approach first estimates the decay rates of neighbouring agents from their safety models and then adjusts the ego agent's decay rate accordingly. However, [27] use the safety model information only to estimate the decay rate of other agents. In this work, we extend beyond previous strategies by explicitly leveraging the safety models of neighbouring agents to directly inform and shape the control actions of the ego agent to improve overall performance.

2.3 Preliminaries

We introduce preliminaries about the system dynamics and Control Barrier Functions in the multi-agent setup.

Definition. (Discrete-time control barrier function): Given a set $C \subseteq \mathbb{R}^n$ defined by (2), the continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ is a *discrete-time control barrier function* for dynamical system (1) if for all $s_t \in C$, there exists $\alpha \in [0, 1]$ such that

$$\sup_{a_t \in \mathcal{A}} \left[h\left(f(s_t, a_t)\right) - (1 - \alpha) h(s_t) \right] \geq 0$$

For any potentially unsafe nominal action a_t^{nom}, we can obtain a safe action solving the quadratic program (QP):

$$a_t = \arg\min_{a_t \in \mathcal{A}} \|a_t - a_t^{\text{nom}}\|_2^2$$

subject to

$$h(f(s_t, a_t)) - (1 - \alpha) h(s_t) \geq 0.$$

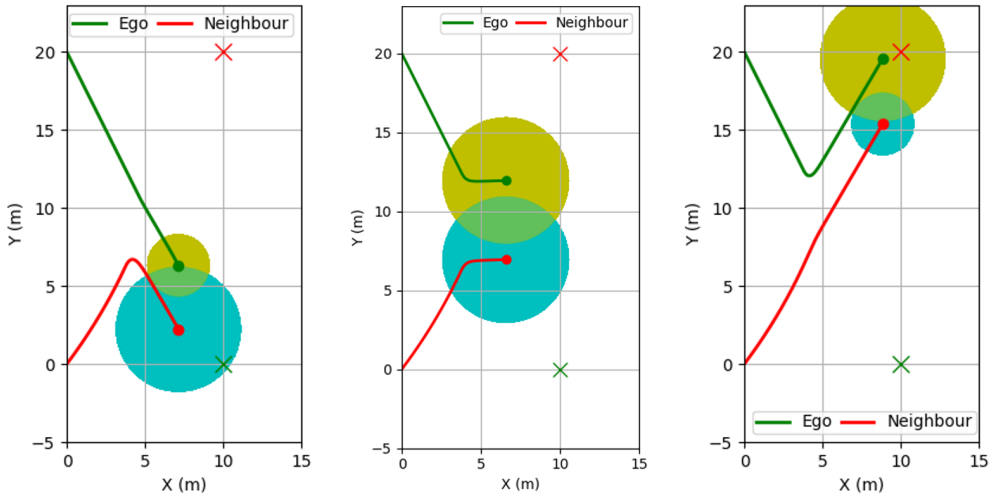


Figure 2.1: Different trajectories for different levels of relative conservativeness of safety of the agents for a given nominal policy.

System dynamics: Consider a multi-agent system with N agents $\mathcal{A} = \{A_i\}_{i=1}^{N}$ in a 2-D environment where every agent i follows the dynamics:

$$\dot{x}_i = f(x_i) + g(x_i)u_i, \quad (1)$$

where $x_i \in \mathbb{R}^n$ denotes the state, $u_i \in \mathbb{R}^m$ the control input, $\dot{x}_i$ the derivative of x_i w.r.t. time t, and $f(x_i), g(x_i)$ the flow vectors for $i = 1, \ldots, N$. Each agent A_i has access to the the states of the other agents $\{x_j\}_{j\in\mathcal{N}_i}$ and its control barrier function $\{h_j\}_{j\in\mathcal{N}_i}$.

2.3.1 Motivation

Consider a multi-agent setup, as illustrated in Fig. 2.1, where safe agents with varying levels of conservativeness interact while navigating toward their respective goal positions. These agents exhibit significantly different trajectories due to variations in their safety boundaries. While safety is maintained during all interactions, the individual trajectories often experience substantial deviations because their nominal policies do not account for the safety models of neighbouring agents. This observation motivates our work, where we aim to leverage the safety models of neighbouring agents to enhance the overall system

performance. Throughout this thesis, the term co-safe behaviour of the ego agent refers to the behaviour that guarantees safety while minimizing its impact on the trajectories of neighbouring agents.

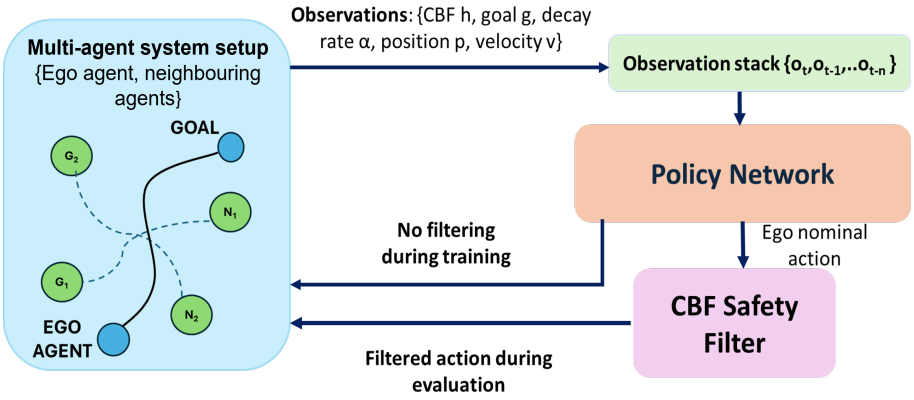


Figure 2.2: Block diagram for the proposed approach.

2.4 Methodology

We consider the reinforcement learning (RL) setup, where an agent interacts with an environment, following the Markov Decision Process (MDP) framework. The MDP is formally defined by the tuple $(\mathcal{O}, \mathcal{A}, p, p_0, \mathcal{R}, \gamma)$, where $\mathcal{O}$ represents the observation space, $\mathcal{A}$ the action space, p the state transition probabilities, p_0 the initial state distribution, $\mathcal{R}$ the rewards, and γ the discount factor. Fig 2.2 illustrates the block diagram of the policy network and the environment, which includes the ego agent and its neighbouring agents. These agents navigate towards their respective goal locations while maintaining a minimum safety distance from each other (shown as circles).

2.4.1 Observation Space and Action Space

In an environment with N agents, the control policy for the ego agent is trained using PPO [40], where both the actor and critic use the same observation space. The observation vector $\mathbf{o}_e$ for the ego agent is as follows:

$$\mathbf{o}_e = \{ p_e,\, v_e,\, g_e,\, h_e,\, \alpha_e,\, \{ p_j,\, v_j,\, g_j,\, h_j,\, \alpha_j \}_{j \in \mathcal{N}_o} \},$$

where $\mathcal{N}_o$ is the set of the nearest m agents. To mitigate the explosion of the observation space as the total number of agents grows, the ego agent restricts its observations from its m nearest neighbours. Here $p_i \in \mathbb{R}^2$ denotes agent i's position, $v_i \in \mathbb{R}^2$ its velocity, $g_i \in \mathbb{R}^2$ its goal, $h_i \in \mathbb{R}$ the instantaneous CBF value, and $\alpha_i \in \mathbb{R}$ the corresponding CBF decay rate. The subscripts e and o denote the corresponding values for the ego agent and the other/neighbouring agents respectively. Although we assume in our analysis that the true decay rate of agents is known to us, it can also be estimated by following a similar method to that in [27]. The corresponding ridge regression formulation is given by:

$$\hat{\alpha}_o = \arg\min_{\alpha} \sum_{i=1}^{n} \|h_o(f(x_k, a_k)) - (1-\alpha)h_o(x_k)\|_2^2 \tag{2.1}$$

where h_o denotes the CBF function for the neighbouring agent. As shown in Fig 2.2, we consider the past n observations as part of the observation stack. For numerical stability, the observations were normalized between $[0, 1]$.

For our analysis, we assume that all the agents follow double integrator dynamics discussed in detail in 2.5.2, therefore the control policy directly outputs the control input for the ego agent, the action vector $\mathbf{a}_e = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$, where a_x and a_y denote the acceleration in the x and y directions respectively.

2.4.2 Assumptions and Approximations

In our overall analysis, we assume that all the agents follow a constant-velocity model. This is a standard assumption made in many multi-agent system research works [8]. The core idea behind this assumption is that we do not have access to other agents' control input and therefore other agents are assumed to operate with constant velocity per time step as follows:

$$v_{k+1} = v_k \quad \Longrightarrow \quad p_{k+1} = p_k + \Delta t\, v_k \tag{2.2}$$

This assumption is utilized by every agent for their CBF calculation (including the ego and neighbouring agents).

An approximation we make in our analysis is an extension of our previous assumption

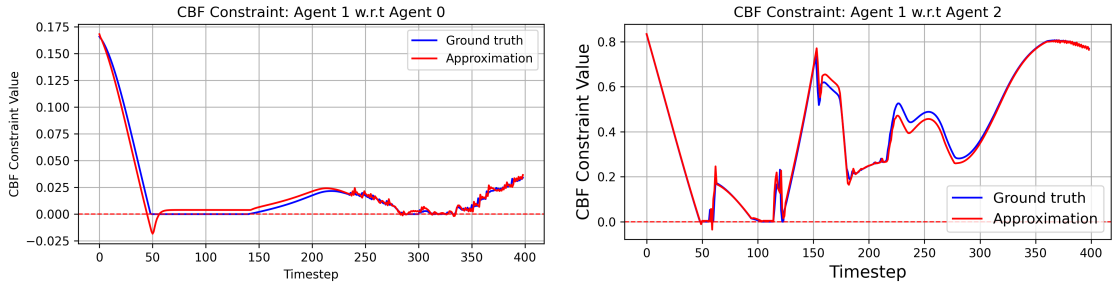


Figure 2.3: CBF constraint values for Agent 1 with respect to its neighbours. (a) Constraint w.r.t. Agent 0; (b) Constraint w.r.t. Agent 2. Blue curves show the ground-truth CBF constraint values, and red curves show their approximations over the 400-step rollout.

that we do not have access to other agents' control inputs. This approximation is utilized for the reward formulation 2.4.3. In continuous-time CBF formulations, calculating the CBF constraint condition requires computing the time derivative of the barrier function h, which depends explicitly on the neighbouring agent's control input a:

$$\dot{h}(x, a) + \alpha h(x) \tag{2.3}$$

[32] utilizes an approximation which lets us calculate the CBF constraint condition without needing control input of the neighbouring agents as follows:

$$\dot{h}(x, a) + \alpha h(x) \approx \frac{\partial h}{\partial s} \left(s_t - s_{t-1}\right) / \Delta t + \alpha h(x)$$

Since we utilize discrete time CBFs, the approximation for discrete time version can be written as follows:

$$h(f(x_k, a_k)) - (1 - \alpha) h(x_k) \approx h(x_k) - (1 - \alpha) h(x_{k-1}) \tag{2.4}$$

Fig. 2.3 shows the ground truth and approximations of the CBF constraint condition. We see a max mean error of 0.001 among different agent constraint conditions and a max error of 0.02 across all timesteps. Since we employ the CBF constraint only for reward shaping and not as a hard safety filter, the observed approximation errors remain well within acceptable limits for our purpose.

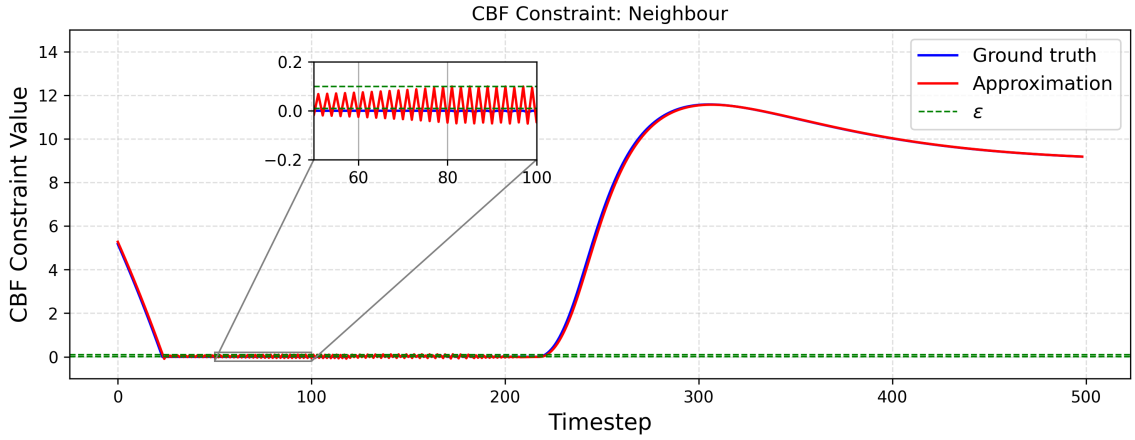


Figure 2.4: Threshold ϵ for Co-safety reward

2.4.3 Reward formulation

The reward design is crafted to promote co-safety behaviour while simultaneously encouraging the agent to achieve the goal-reaching task. At each timestep, the reward r_t is given by

$$r_t = r_t^{\text{prog}} + r_t^{\text{safe}} + r_t^{\text{co-safe}},$$

where the individual rewards are defined as follows:

$$r_t^{\text{prog}} = \lambda_1(d_{t-1} - d_t)$$

$$r_t^{\text{safe}} = \begin{cases} h_{k+1} - (1-\alpha)h_k, & \text{if } h_{k+1} - (1-\alpha)h_k < 0 \\ 0, & \text{otherwise.} \end{cases}$$

$$r_t^{\text{co-safe}} = \begin{cases} -1.0, & \text{if } h_k - (1-\alpha)h_{k-1} < \varepsilon \\ 0, & \text{otherwise.} \end{cases}$$

d_t denotes the distance between the agent and the goal at time t, and λ_1 is a scaling factor for the progress reward. The r_t^{safe} reward ensures that the ego agent respects its own safety boundaries and is penalized only when they are violated. The reward $r_t^{\text{co-safe}}$ uses the CBF constraint condition of the other agent and the goal is to heavily penalize the agent when its constraint condition is activated.

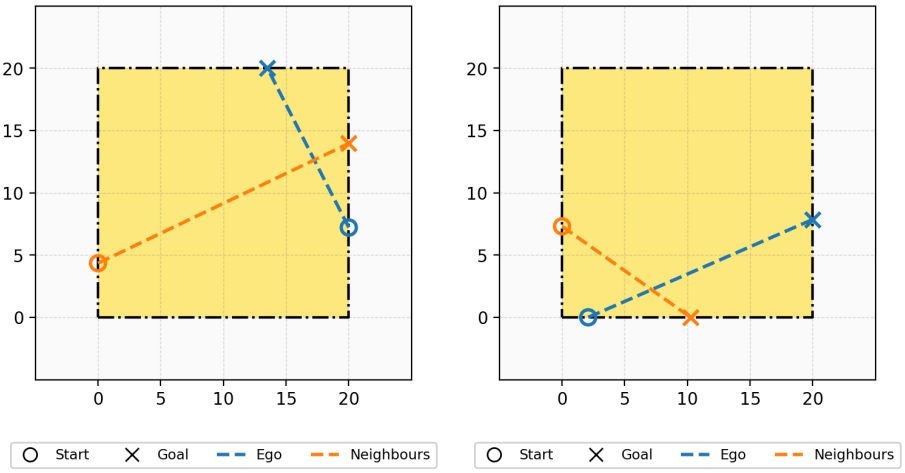


Figure 2.5: Initialization of agents on a 20x20 grid during training.

2.4.3.1 Threshold ϵ for Co-safety reward

The threshold ϵ for the co-safety reward is chosen such that the ego agent is only penalized when the CBF constraint conditions become active for the neigbouring agents due to ego agents' actions. Choosing $\epsilon >> 0$ would encourage the ego agent to always move away from the neighbouring agents. If we had access to the ground truth CBF constraint condition the ideal value of ϵ should be 0 but because we utilize approximate CBF constraint values, the threshold is chosen such that the approximations do not affect the penalization and the chosen value in this case is $\epsilon = 0.1$ as shown in Fig. 2.4.

2.4.4 Domain Randomization

To enhance the generalization capability of the ego agent across diverse scenarios and mitigate the risk of overfitting, we introduce randomization in the simulation environment parameters. In the multi-agent setup, we randomize the minimum safety distance that each agent maintains with the others, denoted as D_s. Additionally, we randomize both the initial and final goal positions on a rectangular border, ensuring that the line connecting the initial and final positions for different agents intersects at some point as shown in Fig. 2.5. This randomization strategy is designed to maximize agent interaction, which accelerates the agent's learning of co-safe behaviours. Furthermore, we randomize the control barrier function (CBF) decay rate for each agent, thereby enhancing the agent's robustness to

varying levels of conservatism in safety.

2.4.5 Network Architecture and Policy Training

We adopt a decentralized multi-agent control framework, where the actions of other agents are not controllable, and control is only applied to the ego agent. The ego agent is trained using Proximal Policy Optimization (PPO), with both the actor and critic networks consisting of a three-layer multi-layer perceptron (MLP) with layer dimensions of [256, 256, 256]. During training, the ego agent's actions are executed without filtering. In contrast, for the other agents, we employ a PD-CBF (Proportional-Derivative Control Barrier Function) policy.

For training, we use curriculum learning with 2 stages: training the agent with only goal reaching reward r^{goal} with domain randomization. In the second stage, we introduce the r^{safe} and $r^{\text{co-safe}}$ rewards to encourage co-safe behaviour once it has learned the goal reaching task. The weight λ_1 of the goal reaching reward r^{goal} is set to 1 and 10 respectively for the first and second stage respectively.

2.5 Experiments

We evaluate our policy in a homogeneous environment, where all agents share identical dynamics and control barrier functions. Our experiments show that when the ego agent uses CBF information of the neighbouring agents, it performs significantly better on the defined metrics compared to the case where the ego agent lacks access to this information.

2.5.1 Metrics

The performance of the multi-agent system is evaluated using the following two metrics:

1. **Dynamic Time Warping (DTW)**: Co-safety behaviour is evaluated by comparing the differences in an agent's trajectory for the goal-reaching task when interacting with other agents (with safety boundaries) to the case when no other agents are present. To quantify this deviation, we use the Dynamic Time Warping (DTW) score [39], which measures the alignment between the observed trajectory and a reference trajectory. The reference trajectory represents the agent's path in the absence of other agents. DTW

is particularly suitable for evaluating deviations, as it operates on discrete data points and ensures that the calculated score is invariant to temporal misalignments within the trajectories. A lower DTW score indicates greater similarity to the reference trajectory, reflecting better co-safety adherence.

2. **Time of Completion**: This metric measures the total task completion time for all agents, with the goal of minimizing it. Our proposed formulation aims to achieve this by fostering cooperative behaviour, where the ego agent adapts its trajectory to influence others courteously, thereby improving overall system efficiency.

2.5.2 Simulation Experiments

We assume that all agents follow the double integrator dynamics defined as follows:

$$\dot{s} = \begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ a \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \begin{bmatrix} 0 \\ a \end{bmatrix} \tag{2.5}$$

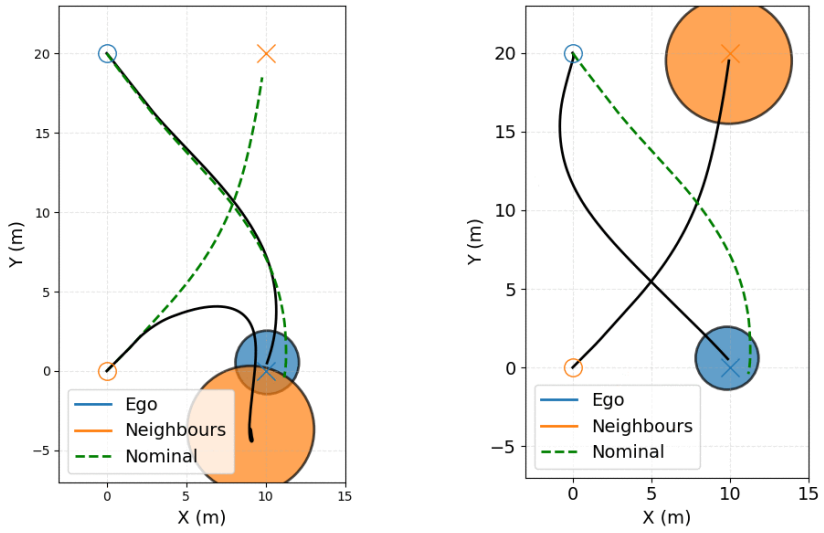
where $p \in \mathbb{R}^2$, $v \in \mathbb{R}^2$, $a \in \mathbb{R}^2$ denote the position, velocity, and acceleration of the agent, respectively. Inspired by [1], we consider the CBFs as pairwise safety constraints between the ego agent i and any other agent $j \neq i$:

$$h(x) = \frac{\Delta p_{ij}^T}{\|\Delta p_{ij}\|} \Delta v_{ij} + \sqrt{a_{\max}\big(\|\Delta p_{ij}\| - D_s\big)}$$

where D_s is the minimum safe distance the agent must maintain from others, $a_{\max}$ denotes the maximum deceleration that the ego agent can apply to prevent collision, and Δp_{ij} and Δv_{ij} denote the relative position and relative velocity.

We train each stage of the policy for 5M steps before transitioning to the next stage. During domain randomization, we vary several parameters: the CBF decay rate α is sampled from the range $[0.1, 1]$, the safety distance D_s from $[2, 5]$ meters, and the initial and goal positions' x and y-coordinates are randomized within $[0, 20]$ meters.

Fig. 2.6 and Fig. 2.7 shows two agents navigating to their goal positions with random initial (denoted by hollow circles) and final positions (denoted by crosses). The orange circles indicate the minimum safety distance that neighbouring agents maintain from others,



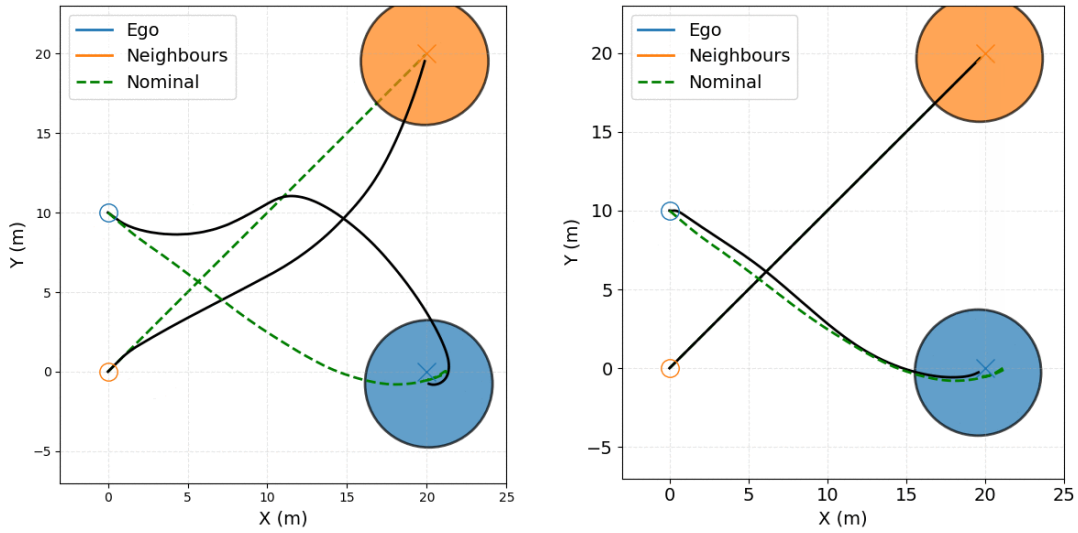
(a) **DTW** = 7384.42, **ToC**= 7384.42 (b) **DTW** = 2524.36, **ToC** = 14.6 s

Figure 2.6: Comparison of agent trajectories under two conditions: (a) the ego agent's path without access to neighbouring agents' CBF information; (b) the ego agent's path with access to neighbouring agents' CBF information. The green trajectory represents an agent's nominal path when operating alone, i.e., without the presence of any other agents.

while the blue circles represent the minimum safety distance for the ego agent. Fig. 2.6 and Fig. 2.7 denote the scenarios when the agents' CBF constraint conditions are activated and we see that total DTW score and total time of completion are significantly reduced using our proposed approach. The corresponding animations for the experiments can be found here.

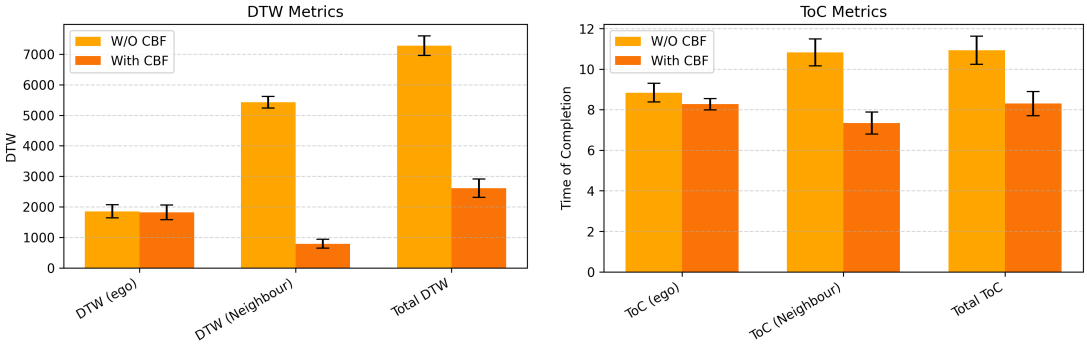
Fig. 2.8 denotes the average individual and total DTW and ToC scores when initializing the agents for random goal and random safety distance when running the experiments across 20 trials. On average we see a reduction of total DTW score by 64.1% and total ToC by 24.1%.

Similarly Fig. 2.9 shows a 4 agent setup where Fig. 2.9 (a) denotes the agents' trajectories with a policy trained only for goal reaching whose actions are filtered using CBF. Fig. 2.9 (b) shows the agents' trajectories with our proposed approach of using neighbouring agents' CBF information. Fig. 2.10 denotes the average individual and total DTW and ToC scores for 4 agent setup when initializing the agents for random goal and random safety distance when running the experiments across 20 trials. On average we see a



(a) **DTW** = 3030.52 **ToC** = 16.5 s (b) **DTW** = 289.3 **ToC** = 14.74 s

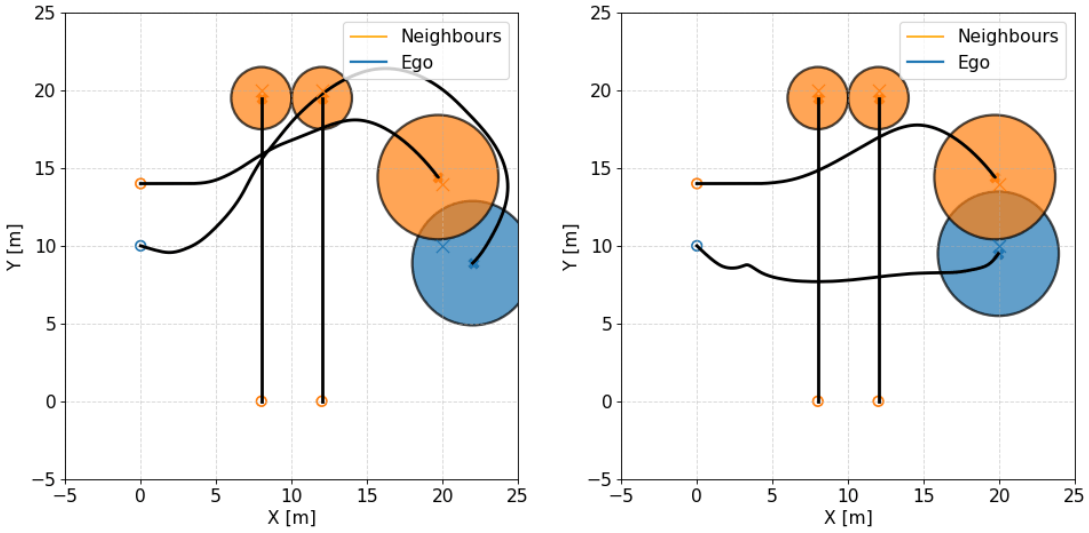
Figure 2.7: Comparison of agent trajectories under two conditions: (a) the ego agent's path without access to neighbouring agents' CBF information; (b) the ego agent's path with access to neighbouring agents' CBF information. The green trajectory represents an agent's nominal path when operating alone, i.e., without the presence of any other agents.



(a) (b)

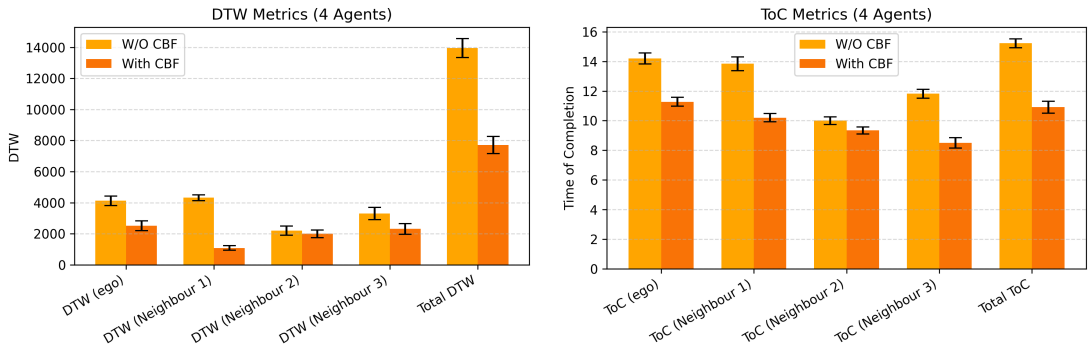
Figure 2.8: (a) Dynamic Time Warping Score metrics (b) Time of Completion metrics for 20 random trials for varying minimum distance, decay rate and initial/goal positions in a 2 agent setup.

reduction of total DTW score by 44.7% and total ToC by 28.4%.



(a) **DTW** = 6310.42, **ToC** = 15.3 s (b) **DTW** = 3389.18, **ToC** = 11.1 s

Figure 2.9: Comparison of agent trajectories under two conditions: (a) the ego agent's path without access to neighbouring agents' CBF information; (b) the ego agent's path with access to neighbouring agents' CBF information.



(a) (b)

Figure 2.10: (a) Dynamic Time Warping Score metrics (b) Time of Completion metrics for 20 random trials for varying minimum distance, decay rate and initial/goal positions in a 4 agent setup.

2.6 Discussion

In this chapter, we introduced a co-safe reinforcement learning framework designed to enhance multi-agent collaboration by leveraging neighbouring agents' CBF information. By explicitly utilizing this information, the ego agent can anticipate potential conflicts, adjust its trajectory proactively, and minimize disruptive interactions. This leads to more courteous behaviour, reducing deviations from the nominal trajectory while maintaining safety.

Empirical results highlight the effectiveness of this approach: our framework achieves a 64.1% reduction in total Dynamic Time Warping (DTW) score and a 24.1% decrease in overall task completion time (ToC) in the two-agent setting. For scenarios involving four agents, we observe a 44.7% reduction in total DTW and a 28.4% reduction in ToC, demonstrating that the proposed method scales effectively to moderately larger teams. These results clearly show that exploiting neighbouring agents' CBF information leads to smoother, more efficient multi-agent coordination.

However, these gains come with certain simplifying assumptions. Our framework assumes that neighbouring agents follow a constant-velocity model. Furthermore, the current work primarily considers homogeneous agents and first-order CBFs, which limits generality in highly diverse environments.

Chapter 3

Model-free Safety Filter for Soft Robots

3.1 Introduction

In the previous chapter, we were focused on improving collective task performance of a system assuming we had access to its accurate model. But often system dynamics are complex and non-rigid, for instance, in soft robots. The question then arises how do we ensure the safety of such systems? Ensuring the safety of soft robots is critical in real-world applications. For example, a soft robotic limb assisting in surgery must avoid sensitive areas of the body to prevent harm. Moreover, excessive bending can overheat the shape memory alloy actuators used in soft robots, leading to permanent damage. To mitigate this, one can define regions of the state space—such as large bending angles—as *unsafe* and avoid them. In this section, we introduce model-free reinforcement learning safety filter design for ensuring safety of soft robots by using simple modifications to standard Soft Actor Critic (SAC) and Q-learning algorithms that allow the agent to learn safe and unsafe regions during training. The designed safety filter is able to reject actions that might lead the system into unsafe states, improving both hardware safety and task reliability. [1]

[1]This section is a collaboration with Andrew Sue as he helped implement half of the software required to realize the proposed method as well as the hardware experiments. The soft robot limb used for experiments was developed by Richard Desatnik. This section is also adapted from our paper [42]

3.2 Related Work

Soft robots are often perceived as inherently safe due to their compliant and deformable structures. However, in medical applications, for instance, many membranes, and nerves are extremely delicate, and even contact with a compliant robotic system can cause damage. These regions must therefore be considered "unsafe" for interaction. Similarly, in shape memory actuated soft robot shown in Fig. 3.6, certain configurations can lead to excessive bending, which risks damaging its actuators. These, too, represent unsafe states. Consequently, ensuring safety in the motion planning and control of soft robots is far from trivial. Despite its importance, research specifically focused on safe control in soft robotics has been relatively limited, with only a few pioneering efforts [34, 38]. Nevertheless, reinforcement learning (RL) has emerged as a promising approach for ensuring safety in high-dimensional, nonlinear, and hard-to-model robotic systems. Since soft robots share these characteristics and are particularly challenging to model accurately, RL methods are a natural choice for their safe control. Building on these developments, existing works on safe RL can be broadly divided into two categories: *model-free safe reinforcement learning*, which enforces safety without relying on an explicit dynamics model, and *model-based safe reinforcement learning*, which leverages system models or learned approximations to provide stronger safety guarantees.

Model-free Safe Reinforcement Learning: Model-free Safe Reinforcement Learning (MFRL) is frequently formulated as a Constrained Markov Decision Process (CMDP) [3]. A CMDP extends a standard MDP by incorporating an additional cost signal to identify state-action pairs that violate constraints. By defining cost thresholds, it is possible to derive policies with low failure probabilities. Popular methods for solving CMDPs include Lagrange multiplier methods [2, 13, 46], projection methods [50], and penalty function methods [18].

Another line of work in model-free RL focuses on learning a safety critic to filter out unsafe actions and is also the inspiration behind our proposed approach. Safety Q-functions for Reinforcement Learning (SQRL) [41] learns a safety critic that predict the future failure probability and uses the critic to constrain the nominal policy. The framework involves pre-training the safety critic and then fine-tuning the policy on target tasks using the learned safety precautions. Expanding on this approach, Recovery RL [47] additionally learns a recovery policy along with safety critic. They utilize two separate policies for task and

recovery to learn safely without compromising task performance.

In contrast to [47], [20] focuses on jointly optimizing performance and safety. They modify the reward design proposed in [48] by incorporating the safety critic into the reward to prevent exploration in unsafe regions during training. [9] introduces a binary safety critic by leveraging the idea that safety is a binary property.

Model-based Safe Reinforcement Learning: Model-based safe RL methods are more sample efficient compared to model-free methods. Recent studies integrate CMDPs with model-based RL to minimize training violations and speed up learning [6, 48, 51]. Safe Model-Based Policy Optimization (SMBPO) [48] utilizes the learned model for planning to prevent safety violations by penalizing unsafe trajectories. Their approach, however assumes safety violations occur within a given horizon after entering irrecoverable states. Moreover, uncertainties in the learned model can lead to incorrect predictions, causing the agent to misclassify unsafe states as safe.

In contrast, the approach proposed in this work does not rely on assumptions about the time horizon of irrecoverable states. Instead, it adopts a model-free strategy to ensure safety, thereby avoiding issues related to model inaccuracies and offering a more robust safety framework.

3.3 Preliminaries

3.3.1 State Space Partitioning

Consider a car traveling along a road (Fig. 3.1) where areas like the curb are designated as unsafe states ($\mathcal{X}_{\text{unsafe}}$) due to hazards. If the car is traveling at $80\,\text{km/h}$ and is only $1\,\text{m}$ from the curb, a collision is inevitable even with maximum deceleration, defining this state as the irrecoverable state ($\mathcal{X}_{\text{irrec}}$). In contrast, if the car is $1\,\text{km}$ away from the curb, there are various control actions that can ensure safety, categorizing this state as an absolutely safe state ($\mathcal{X}_{\text{safe}}$). Therefore, we can divide the state space $\mathcal{X}$ into three subspaces: $\mathcal{X}_{\text{safe}}$, $\mathcal{X}_{\text{irrec}}$ and $\mathcal{X}_{\text{unsafe}}$.

Identifying irrecoverable states is challenging due to system dynamics, but crucial, since entering $\mathcal{X}_{\text{irrec}}$ leads to $\mathcal{X}_{\text{unsafe}}$.

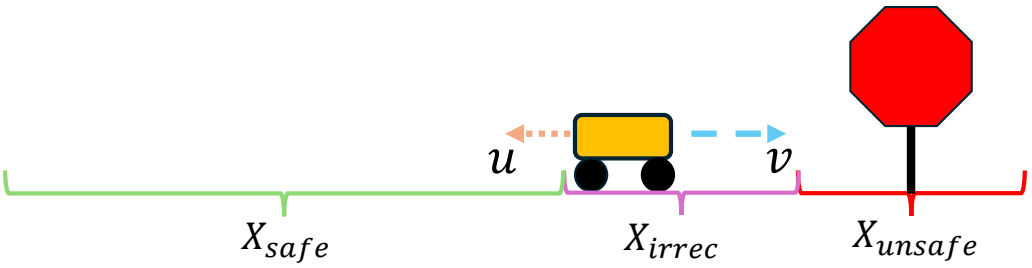


Figure 3.1: The full state space is broken down into three regions, $\mathcal{X}_{safe}$, $\mathcal{X}_{irrec}$, $\mathcal{X}_{unsafe}$. $\mathcal{X}_{safe}$ is the region in which the control input u can always be applied to prevent the system from entering $\mathcal{X}_{unsafe}$. $\mathcal{X}_{irrec}$ is the region where no control input can prevent entry into $\mathcal{X}_{unsafe}$. If a car is moving too fast and too close to the unsafe region, it is an example of the system being in the irrecoverable region.

3.3.2 Q Learning

We propose to use Q-learning for constructing a safety filter that filters hazardous actions and keeps the system in $\mathcal{X}_{safe}$. In Q-learning, the goal is to learn the expected discounted cumulative reward for a state-action pair in a Markov Decision Process (MDP) [43]. The Q-value is updated during training using the Bellman update rule, under deterministic dynamics:

$$Q(x,u) = r(x,u,x') + \gamma \max_{u'} Q(x',u') \tag{3.1}$$

where r is the reward function and γ is the discount factor. The value function is defined as:

$$V(x) = \max_{u} Q(x,u) \tag{3.2}$$

We propose learning optimal Q-function and value function Q^*_{safe} and V^*_{safe}, respectively, using a specialized reward function r_{safe}, where states in $\mathcal{X}_{safe}$ exceed a threshold ϵ_2. This threshold helps in determining whether actions from task-specific nominal policies π_{task} are unsafe based on the learned Q and V functions, $\hat{Q}_{safe}$ and $\hat{V}_{safe}$, and replacing them with safer actions that provide the highest expected discounted cumulative safe reward.

3.4 Methodology

In this work, we propose a Q-learning-based, model-free reinforcement learning approach to construct a safety filter. A block diagram of the framework is shown in Fig. 3.2. Our

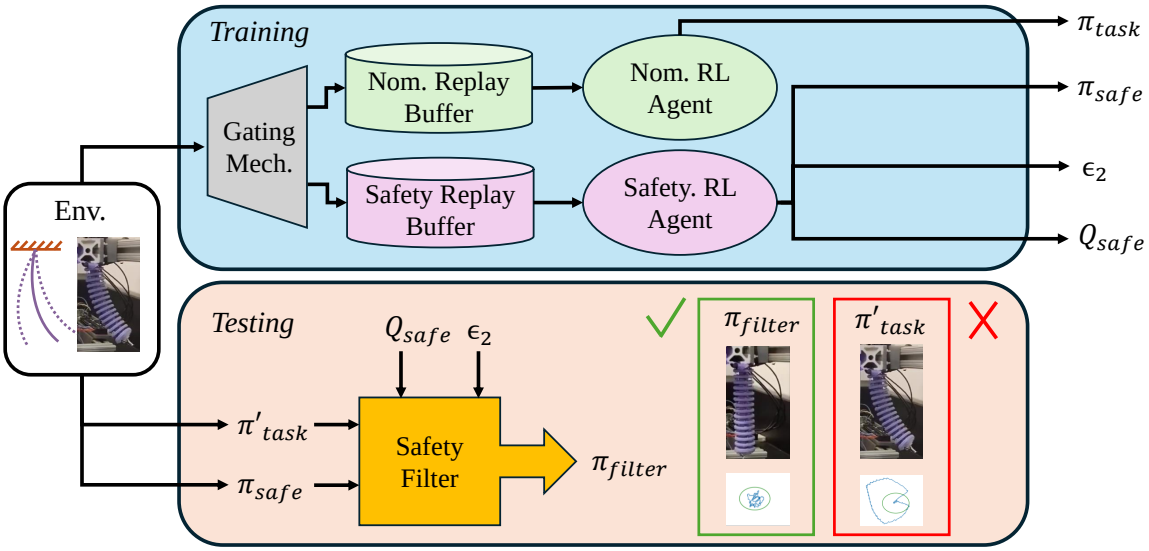


Figure 3.2: The block diagram shows our model-free RL-based safety filter framework. During training, environment observations are stored in the replay buffers of both task-specific nominal and safety agents, enabling their simultaneous training. In testing, observations are processed by both policies, and the nominal action is filtered based on the safety agent's Q-function and threshold ϵ_2.

method leverages the fact that $\hat{Q}_{safe}$ and $\hat{V}_{safe}$ can be learned off-policy, enabling the simultaneous training of a task-specific policy π_{task} and the safety policy π_{safe}.

The two policies are trained in parallel but remain decoupled by a gating mechanism that sorts observations into separate replay buffers based on episodic conditions. This decoupling allows π_{task} to be swapped with any π'_{task} during testing. Additionally, a gradient-guided search is used to optimize ϵ_2 for π_{filter}, ensuring good performance based on the learned $\hat{V}_{safe}$.

3.4.1 Reward Formulation

We define the safety reward function $r_{\text{safe}}(x, u, x')$ as:

$$r_{\text{safe}}(x,u,x') = \begin{cases} l(x), & x, x' \notin \mathcal{X}_{\text{unsafe}} \\ -\dfrac{1}{\gamma^t(1-\gamma)}, & x \notin \mathcal{X}_{\text{unsafe}},\ x' \in \mathcal{X}_{\text{unsafe}} \\ -1, & x, x' \in \mathcal{X}_{\text{unsafe}} \end{cases} \tag{3.3}$$

where x is the state, comprised of time t, position and velocity. x' is the state the system transitions to after applying control u, and $\gamma \in (0, 1)$ is the discount factor. The function $l(x) \in (0, 1]$ increases as the system moves deeper into the safe region; for example:

$$l(x) = \frac{d(x)}{\max_{x \notin \mathcal{X}_{\text{unsafe}}} d(x)}, \tag{3.4}$$

where $d(x)$ is the positive signed distance to $\mathcal{X}_{\text{unsafe}}$. For x where $d(x) = 0$, i.e., on the boundary of $\mathcal{X}_{unsafe}$, we consider $x \in \mathcal{X}_{unsafe}$. For incorporating N safety constraints into the reward design, the distance is defined as $d(x) = \min\{d_1, d_2, ...d_N\}$ where d_i represents the positive signed distance to constraint i. Furthermore, we assume that the safe set is bounded by either safety or dynamic constraints.

During training, we optimize the $\hat{Q}_{\text{safe}}$ function using standard Bellman updates using r_{safe}.

Furthermore, we assume an episode terminates at the time step when the agent reaches the unsafe region or at maximum episode length, T. We consider a finite episode length scenario, since from Eq. 3.3, when $x \in \mathcal{X}_{\text{safe}}$ and $x' \in \mathcal{X}_{\text{unsafe}}$, the reward can become unbounded if t is very large. Therefore, we define T as the episode length during training to bound the reward and prevent divergence.

Our design of r_{safe} aims to ensure a clear separation on $\hat{V}_{safe}$ between states in $\mathcal{X}_{\text{safe}}$, and those in $\mathcal{X}_{\text{irrec}}$. While an intuitive approach is to add a large negative penalty upon entering the unsafe region [48], fixed penalties diminish over time due to discounting, blurring the value separation between $\mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{irrec}}$. This reward formulation ensures a clear separation between $\mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{irrec}}$ in discounted cumulative rewards shown in the properties described below.

3.4.2 Reward Properties

The true value function V^*_{safe} or the maximum discounted cumulative reward has the following properties.

1. If $x \in \mathcal{X}_{safe}$, then the optimal value of the value function satisfies the following: $0 < V^*_{safe}(x) \leq \frac{1-\gamma^{T+1}}{1-\gamma}$.
2. If $x \in \mathcal{X}_{irrec}$, then the optimal value of the value function satisfies the following $-\frac{1+\gamma}{1-\gamma} \leq V^*_{safe}(x) < 0$.

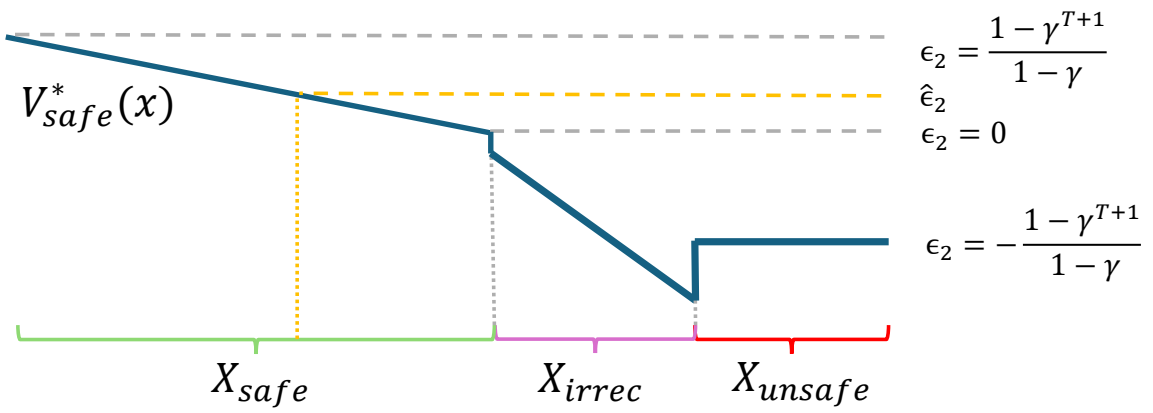


Figure 3.3: A 1-Dimensional concept visualization of what V^*_{safe} could be like. $\epsilon_2 = 0$ is the threshold value that separates $\mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{irrec}}$. Because V^*_{safe} is increasing deeper inside $\mathcal{X}_{\text{safe}}$, picking a threshold value $\hat{\epsilon}_2$ that is higher than the optimal ϵ will give a more conservative estimate of the safe set.

3. If $x \in \mathcal{X}_{unsafe}$, then the optimal value of the value function satisfies the following $V^*_{safe}(x) = -\frac{1-\gamma^{T+1}}{1-\gamma}$.

These properties show that $V^*_{\text{safe}}(x)$ generally increases from $\mathcal{X}_{\text{unsafe}}$ to $\mathcal{X}_{\text{safe}}$, especially the transition from $\mathcal{X}_{\text{irrec}}$ to $\mathcal{X}_{\text{safe}}$. This indicates that the boundary between $\mathcal{X}_{\text{irrec}}$ and $\mathcal{X}_{\text{safe}}$ corresponds to a threshold value distinguishing these two regions in $V^*_{\text{safe}}(x)$. Fig. 3.3 depicts a conceptual visualization of the $V^*_{\text{safe}}(x)$.

From these properties, and the fact that $V^*_{safe}(x) = \max_{\mathbf{u}\in\mathcal{U}} Q^*_{safe}(x,u)$, where $\mathcal{U}$ denotes the set of permissible control inputs, the following action filtering scheme to ensure safety results:

$$\pi_{filter}(x) = \begin{cases} \pi_{task}(x), & Q^*_s(x, \pi_{task}(x)) > \epsilon_2 \\ \underset{u\in\mathcal{U}}{\text{argmax}}\, Q^*_s(x,u), & Q^*_s(x, \pi_{task}(x)) \leq \epsilon_2 \end{cases} \tag{3.5}$$

where ϵ_2 denotes the value threshold that separates $\mathcal{X}_{safe}$ and $\mathcal{X}_{unsafe}$. Q^*_s represents Q^*_{safe}, π_{task} is any policy trained to achieve a specific task under arbitrary task rewards r_{task}, and $\text{argmax}_{\mathbf{u}\in\mathcal{U}} Q^*_s(x,\mathbf{u})$ is the safety policy π_{safe}. π_{filter} refers to the filtered policy that ensures task completion while maintaining safety. For the optimal V^*_{safe}, we have $\epsilon_2 = 0$. Details regarding the theoretical proof and setup can be found in [42].

3.5 Implementation

To implement π_{filter} and keep the system within $\mathcal{S}_{\text{safe}}$, the task policy action is replaced by a greedy policy that maximizes the learned Q_{safe} under r_{safe}. We use Soft Actor-Critic (SAC) [19] for simulations with continuous action spaces and Deep Q-Learning (DQN) [30] for real-world validation.

3.5.1 Simultaneous Training of Task Policy and Safety Policy

Leveraging the off-policy nature of SAC and DQN agents, we simultaneously train a task agent that maximizes task reward and a safety agent that maximizes the proposed safety reward. Both agents share the environment and rolled-out data but maintain independent replay buffers. A gating mechanism prevents the safety agent from collecting observations once the system enters the unsafe region. Initially, observations are added to both agents' replay buffers with their respective rewards. When the safety agent reaches the unsafe region, it stops receiving new observations, while the nominal agent continues until the episode ends. This approach minimizes interference with the task agent, preserving task performance, and ensures the safety policy remains valid for any task policy despite simultaneous training and shared data.

Due to the early termination assumption, the safety agent does not observe the unsafe region, as the episode terminates once the unsafe region is reached. Therefore a supervised loss, similar to the loss used in [44], is introduced to the safety agent in addition to the standard RL losses to classify whether a state belongs to $\mathcal{X}_{unsafe}$. The loss is defined as follows:

$$L_{\text{unsafe}} = \left\| V_{\text{safe}}(x) + \frac{1-\gamma^{T+1}}{1-\gamma} \right\|, \quad \forall\, x \in \mathcal{X}_{\text{unsafe}} \tag{3.6}$$

where V_{safe} denotes the learned value function.

3.5.2 Searching for ϵ_2

Precisely converging to V^*_{safe} is challenging, and while the theoretical optimal value $\epsilon_2 = 0$ holds under ideal conditions, it may not represent the actual boundary for V_{safe}. Assuming the agent is sufficiently trained so that $V_{\text{safe}} \approx V^*_{\text{safe}}$ and has a similar shape, a threshold value separating $\mathcal{S}_{\text{safe}}$ and $\mathcal{X}_{\text{irrec}}$ should exist due to the increasing nature of V^*_{safe} from the

Algorithm 1 Gradient-Based Search for ϵ_2

Initialize x randomly such that $V_{\text{safe}}(x) > 0$
Set step size α, tolerance δ, and max episode length T_{max}
$V_{\text{prev}}, l_{\text{prev}} \leftarrow -\infty$
while $|V_{\text{safe}}(x) - V_{\text{prev}}| > \delta$ **do**
    $V_{\text{prev}} \leftarrow V_{\text{safe}}(x)$
    Roll out the safety policy from state x for one episode
    $l \leftarrow$ length of the episode
    if $l = T_{\text{max}}$ **then**
        if $l_{prev} \neq T_{max}$ **then**
            $\alpha \leftarrow 0.5 * \alpha$
        end if
        $x \leftarrow x - \alpha \nabla V_{\text{safe}}(x)$
    else
        if $l_{prev} = T_{max}$ **then**
            $\alpha \leftarrow 0.5 * \alpha$
        end if
        $x \leftarrow x + \alpha \nabla V_{\text{safe}}(x)$
    end if
    $l_{prev} \leftarrow l$
end while
$\epsilon_2 \leftarrow V_{\text{safe}}(x)$

irrecoverable to the safe region.

To find this boundary ϵ_2 on V_{safe}, we propose a gradient-guided linear search method (Alg. 1). The algorithm determines whether a state x belongs to $\mathcal{S}_{\text{safe}}$ or $\mathcal{X}_{\text{irrec}}$ by rolling out the safety policy $\pi_{\text{safe}} = \arg\max_{\mathbf{u} \in \mathcal{U}} Q^*_{\text{safe}}(x, \mathbf{u})$ and checking if the system survives the maximum episode length T. If the system survives, x is likely in $\mathcal{S}_{\text{safe}}$, suggesting the boundary is less than $V_{\text{safe}}(x)$, prompting the algorithm to descend along the gradient at x. Conversely, if the system enters the unsafe region before T, x is likely in $\mathcal{X}_{\text{irrec}}$, and the algorithm ascends along the gradient to find ϵ_2.

Since V_{safe} is a neural network approximation and may not be smooth, the algorithm might output local optima. Therefore, we repeat the algorithm multiple times with random starting states and select the maximum ϵ_2 from these trials. The resulting ϵ_2 is used as the threshold in π_{filter}.

Due to stochasticity and imperfections in neural network approximations, particularly

overestimation common in DQN, the threshold from Alg. 1 may not correspond to the actual boundary between safe and irrecoverable regions. However, it provides a useful guideline. Moreover, because π_{safe} is not guaranteed to be safe in suboptimal cases, raising ϵ_2 results in more conservative estimates of the safe region. This means π_{filter} will more frequently select actions from the safety policy, reducing the likelihood of reaching irrecoverable states. As shown in the section 3.6, by using the search algorithm's output as a starting point and manually increasing the threshold, our method can keep the system safe even when significant discrepancies exist between the learned and actual value functions.

3.6 Experiments and Results

We experimentally validate our method in simulations and on real hardware. Simulations involve a double integrator system for basic validation and a Dubin's car environment for static obstacle avoidance to compare with existing methods. On real hardware we demonstrated the filter's capability to constrain a soft robotic limb within a specified region.

3.6.1 Simulation Environments

3.6.1.1 Double Integrator

The double integrator state is two-dimensional, with position x and velocity $\dot{x}$. Safe bounds are ± 2 m and ± 3 m/s, with absolute maximum bounds of ± 4 m and ± 5 m/s. The input acceleration is bounded between ± 2 m/s^2, and the task is to reach a goal at 1.8 m (neon green dashed line in Fig. 3.4).

The two-dimensional state allows easy visualization of the learned value function. In Fig. 3.4, safety policy actions (acceleration) are shown on the left, and the value function on the right. The solid black contour represents the analytical safe set [15]. The black dashed lines denote the learned safe set where $\epsilon_2 = 0$, which lies entirely within the analytical safe set. The blue dashed line represents the contour for $\epsilon_2 = 90$; as per our formulation, this contour is smaller than the $\epsilon_2 = 0$ contour, leading to more conservative estimates of $\mathcal{S}_{\text{safe}}$, which is observed empirically.

Moreover, on the left of Fig. 3.4, at the boundary of the analytical safe set, the actions align with intuition: when approaching the right boundary at high speed, the action is

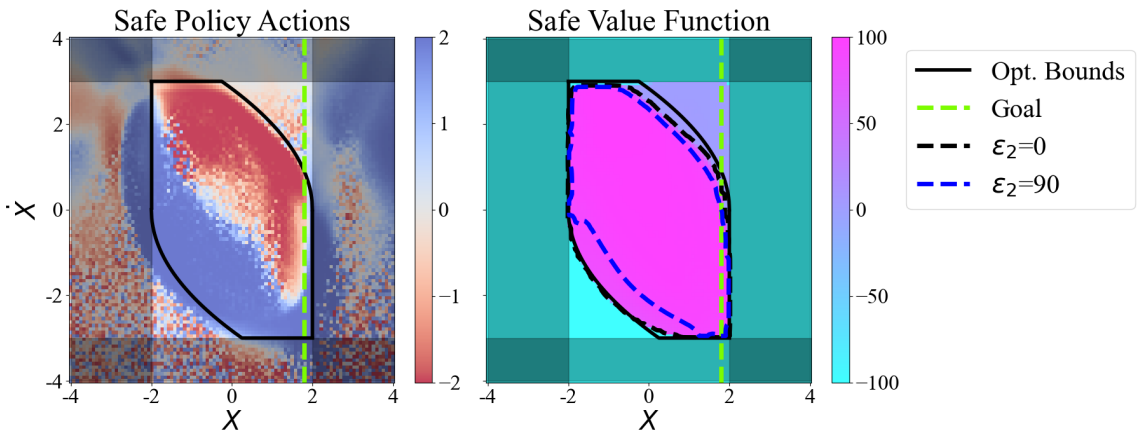


Figure 3.4: *Left*: The actions of the safe policy as a function of the state space. *Right*: The value function of the safety agent for double integrator. The dark shaded area is the unsafe region.

a leftward force (red), and vice versa. These empirical results are consistent with our theoretical expectations.

3.6.1.2 Dubin's car

Because the double integrator's goal lies within the safe region, the nominal agent remains safe by simply achieving the task, which doesn't demonstrate the full effect of our safety filter. Therefore, we tested our method in the Dubin's car environment, featuring complex nonlinear dynamics and conflicts between the nominal task and safety, e.g. the shortest path to goal goes through $\mathcal{X}_{unsafe}$. In this environment, safety bounds are ± 2 m with a central keep-out region of radius 1 m. The absolute maximum bounds are defined with an input velocity of 1.2 m/s, aiming for a goal at $(1.8$ m, 1.8 m$)$ with a radius of 0.5 m. As shown in Tab. 3.1, our method's performance is comparable to, and sometimes better than, other common safe RL algorithms.

In Tab. 3.1, we compare our filtered policy (safety filter with co-trained nominal policy) against Lagrangian Relaxation (LR), Risk Sensitive Policy Optimization (RSPO) [47], the unconstrained task policy, and a task policy trained with a large penalty for entering the unsafe region (Reward Penalty). Results are the mean and standard deviation over 10 runs with different seeds. Using $\epsilon_2 = 67.38$ found by Alg. 1 for our safety threshold, our method attains the highest average episodic return among methods achieving a 100% safety rate

(fraction of episodes reaching the maximum length without safety violations out of 100 episodes).

We also tested our filter with nominal policies not co-trained with our safety agent (Tab. 3.2): PPO [40], DDPG [24], TD3 [16], and a uniform random policy. Over 10 runs with different seeds, only PPO had safety violations, with an average safety rate of 98.8%. We attribute this slightly less than perfect safety rate to the stochastic nature of our safety filter trained with SAC as it samples from a learned distribution to output an action.

As shown in Fig. 3.5, there is a clear correlation between ϵ_2 and performance. Increasing ϵ_2 results in greater safety but reduced nominal rewards, aligning with our theoretical prediction that higher ϵ_2 leads to a more conservative method. The blue cross indicates $\hat{\epsilon}_2$, the value found by Alg. 1.

Table 3.1: Performance comparison of our safety filter to other safe model-free reinforcement learning methods.

	Average Episodic Return	**Safety Rate**
Unconstrained Task Policy	46.38 ± 20.37	0.017 ± 0.051
RSPO	13.37 ± 4.61	0.855 ± 0.138
Reward Penalty	-2.10 ± 8.49	1.000 ± 0.000
LR	2.84 ± 2.34	1.000 ± 0.000
Filtered Policy	***7.08 ± 3.37***	***1.000 ± 0.000***

Table 3.2: Safety Rate of filtering different policies

	Safety Rate	**Std. Safety Rate**
Co-trained Policy	1.0	0.0
PPO	0.988	0.023
DDPG	1.0	0.0
TD3	1.0	0.0
Random	1.0	0.0

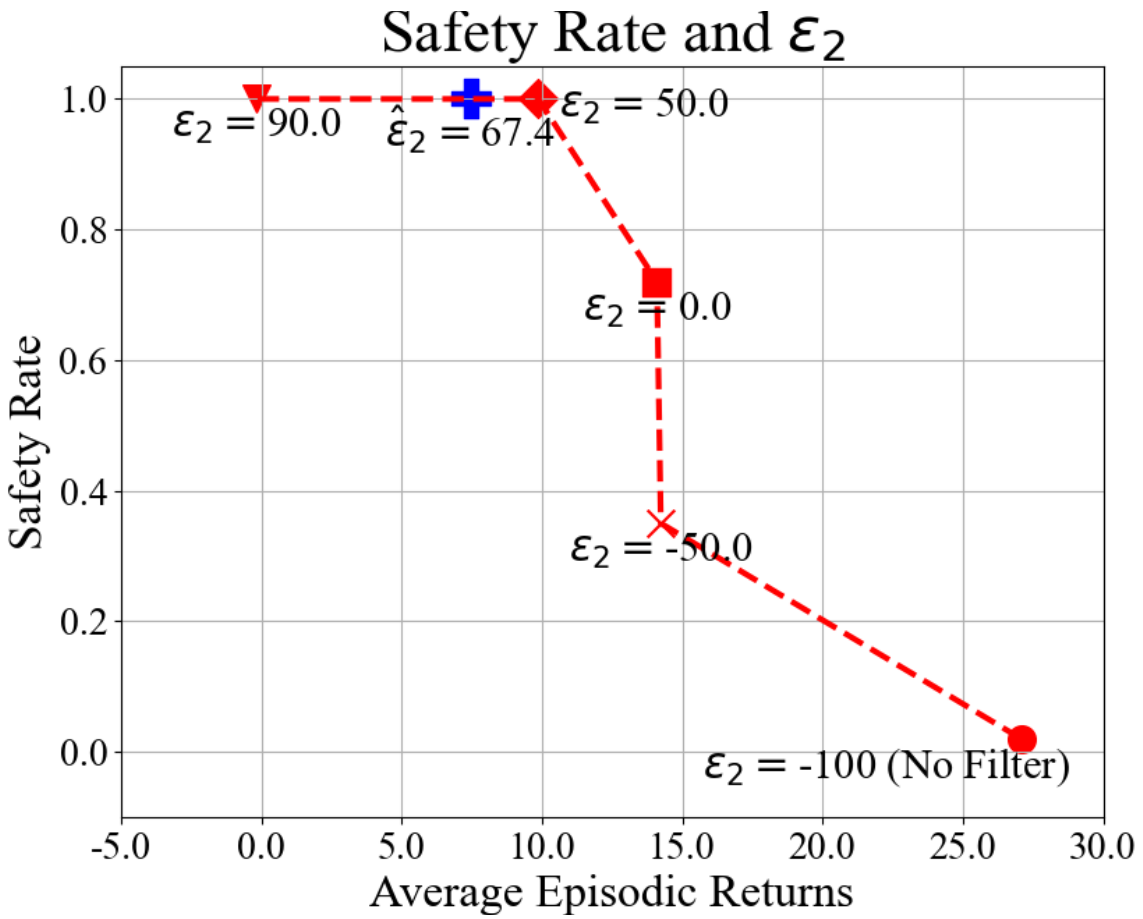


Figure 3.5: Performance of our filter evaluated by average episodic return and safety rate at different ϵ_2 levels in the Dubin's car environment.

3.7 Hardware Experiments

3.7.1 Hardware Setup

To experimentally validate the proposed scheme, we implemented it on a soft robotic limb similar to the setup used in [12] and is depicted in Fig. 3.6. The soft limb is injection-molded from silicone (Smooth-Sil 945) and actuated by four shape memory alloy (SMA) coils (Flextinol) positioned at the up, down, left, and right orientations. The coils contract when heated by a Pulse Width Modulated (PWM) controlled electrical current that are applied at the coil tips. When the temperature rises above $90°\mathrm{C}$, due to the phase transition of the SMA, the limb bend in a specific direction, increasing the bending angle.

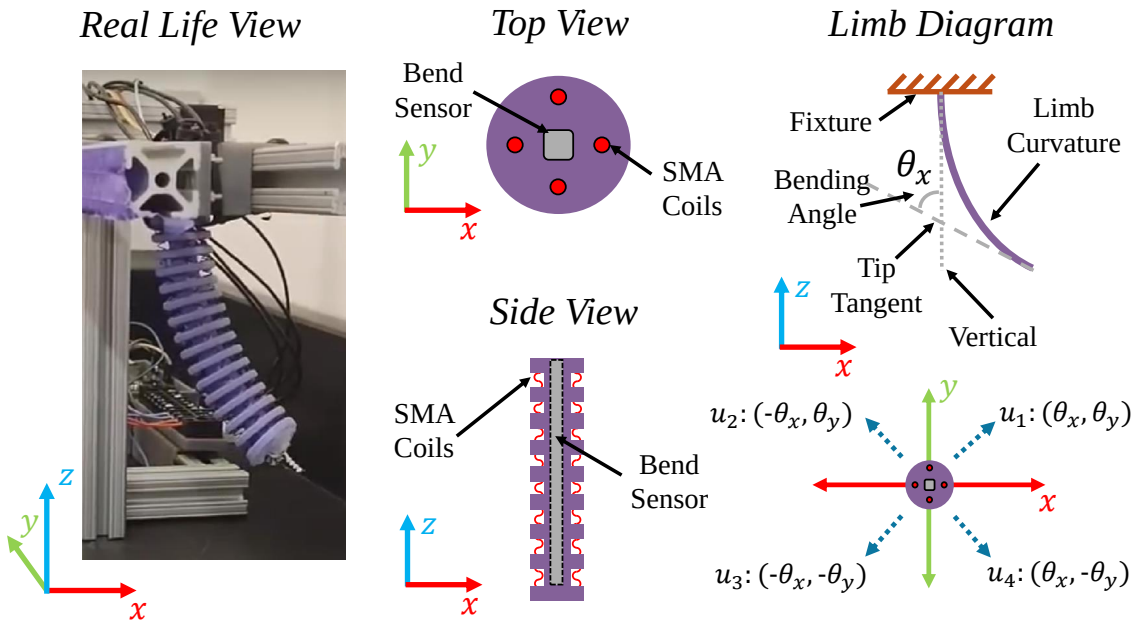


Figure 3.6: Experimental setup of the soft robotic limb. *Left*: Soft limb mounted on an aluminum fixture, actuated by SMA coils controlled via PWM through an Arduino UNO. *Middle*: Cross-sectional view (not to scale) showing the embedded two-axis bending angle sensor (gray) in the silicone limb (purple), with SMA coils at cardinal directions. *Right*: Diagram illustrating the bending angle and the four example actions used.

A capacitive bend sensor (Bend Labs Digital Flex Sensor - 2-Axis, 4 Inch) is located in the center to detect the bending angles of the limb. The bending angles (θ_x, θ_y) are defined as the angle between the tangent at the limb tip and the vertical, measured along both the x and y axes. Since the soft limb's actuation depends on heat—increased heat leads to greater contraction—and due to the malleable nature of silicone, the system exhibits highly nonlinear dynamics. Furthermore, we control only the PWM signals that induce heating to raise the temperature, but relying only on passive cooling to lower it. These factors complicate the dynamics, rendering an analytical mapping from PWM signals to limb states intractable.

3.7.2 RL Setup

To validate our method, we trained the safety policy using DQN with the observation space $x = [\theta_x, \theta_y, \dot{\theta}_x, \dot{\theta}_y]$ and a discrete action space - move upper right (u_1), upper left (u_2), lower

left (u_3), and lower right (u_4)—as depicted in Fig. 3.6. The safety policy was trained using our approximated dynamics simulator to constrain movements within $\pm 30^\circ$ on both x and y axes. The constraint boundary is shown as a green circle in the right figure of Fig. 3.6, i.e., $\mathcal{X}_{\text{safe}}$ represents states where $\|\theta\| < 30^\circ$.

For the nominal policy, we used a predefined action sequence: u_1 for 3 seconds, u_4 for 3 seconds, u_3 for 6 seconds, u_2 for 12 seconds, u_1 for 6 seconds, and u_3 for 3 seconds, aiming to trace a diamond-shaped trajectory. The real-life rollout of this sequence is shown in orange in Fig. 3.7. The task of the safety filter is to constrain the limb movement within the $\|\theta\| < 30^\circ$ boundary.

3.7.3 Hardware Results

As shown in Fig. 3.7, the safety policy successfully keeps the limb within the safe region, as the blue trajectories, which represent the recorded limb trajectory with our safety filter, are entirely contained within the safe region denoted by the green circle. However, due to the sim-to-real gap between the learned dynamics and the real limb dynamics, $\epsilon_2 = 90$ was required for our scheme to work in real life, resulting in a very conservative filter. Despite this conservatism, the empirical results validate the robustness of our method in maintaining system safety, even in the presence of a significant sim-to-real gap.

Additionally, in Fig. 3.8, we observe that as ϵ_2 increases, the system becomes "safer", meaning it takes longer before reaching the unsafe region, aligning with the theoretical predictions and demonstrating the robustness of our approach.

3.8 Discussion

In this section, we propose a reinforcement learning approach that uses reward shaping to distinguish between safe, unsafe, and irrecoverable states. Central to our method is a policy filter that provides theoretical safety guarantees under ideal conditions and can be adapted—through threshold tuning—to perform effectively even in less-than-optimal settings, such as environments with significant sim-to-real discrepancies.

Our framework enables the concurrent training of both a task-specific policy and a safety policy, the latter of which is designed to generalize across various task policies. In simulation benchmarks, our method performs on par with existing approaches and is

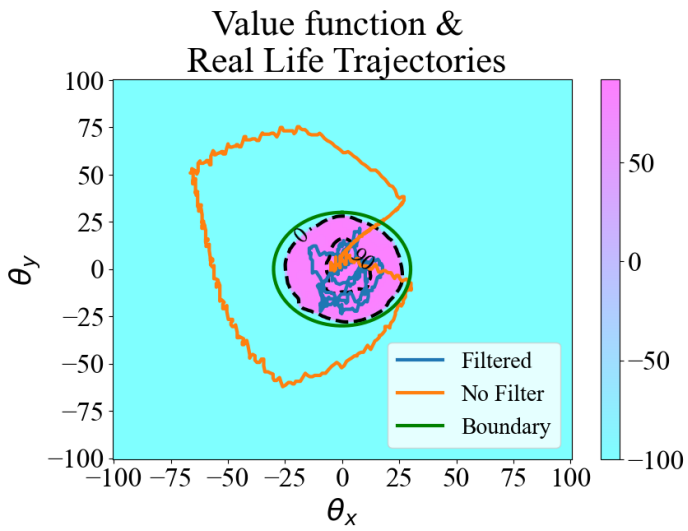


Figure 3.7: The real-life recorded limb trajectory overlaid on the learned value function. Note that the value function is learned through simulation and is a 2D projection of a 4D function, whereas the trajectory values are recorded in real life. The specific value function visualization is generated by setting the velocity terms in the states to 0. The 0 and 90 threshold contour is denoted by dark dashed lines.

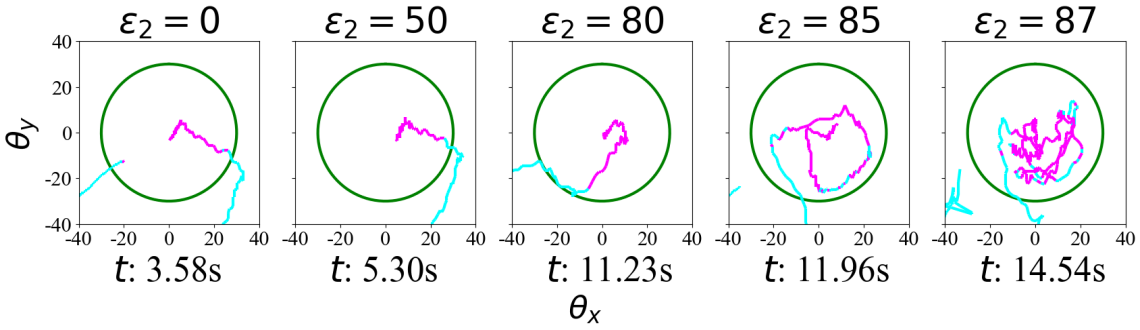


Figure 3.8: Visualization of recorded trajectories for different values of ϵ_2. t denotes the average amount of time before the limb reaches the unsafe region for a total of 5 hardware trials for the specific value of ϵ_2 it is under. The magenta segments denote segments of the trajectory where the value function is above the specific ϵ_2, the cyan denotes segments of trajectory below the specific ϵ_2. Note that the value function computed here uses real velocity information, as well, unlike Fig.3.7

further validated on a real-world, highly nonlinear soft silicone limb. This demonstrates the method's practicality, improved training efficiency, and broader applicability compared to prior work.

Chapter 4

Conclusion

4.1 Limitations and Future Work

Although the proposed framework in Chapter 2 introduces a novel pipeline to improve system performance by leveraging the CBF information of the other agents, we consider a simple setting with only two agents. Future work would involve extending this setting to multiple agents and high-dimensional systems. Moreover, this work mainly focused on the homogeneous setting, i.e., where all agents followed similar dynamics and control barrier functions. In the future, we would like to explore cases in which heterogeneous systems also benefit from this approach. Moreover, our approach necessitates the use of first-order CBFs, whereas in the future we will try to explore how we can accommodate higher-order CBFs into our reward design.

Regarding the safety filter proposed in Chapter 3, while decoupling task and safety policies improves generalization, it can lead to issues in scenarios where safety and task objectives conflict, potentially resulting in deadlock. Future work could investigate safety filter designs that retain generalizability while minimizing task constraints. Moreover, although the proposed approach performs well under suboptimal conditions, it currently lacks formal safety guarantees. Extending the framework to provide such guarantees even when operating under imperfect policies presents an important and impactful avenue for future research.

4.2 Summary

This thesis presents two complementary approaches to enhance safety and performance in robotic systems, focusing on multi-agent environments and soft robots. In multi-agent systems, our co-safe reinforcement learning framework effectively balances collision avoidance with task efficiency by leveraging neighbouring agents' safety models. Simulation studies in both two-agent and four-agent scenarios demonstrate significant improvements in Dynamic Time Warping (DTW) scores and task completion times, validating the effectiveness of our approach.

For soft robotic actuators, our Q-learning-based, model-free safety filter successfully maintains safety in complex, nonlinear dynamics. Our filter can operate alongside any nominal policy, filtering unsafe actions to maintain system safety. We validate the framework in simulation and on a real soft robotic limb, demonstrating its robustness even under notable sim-to-real discrepancies.

Together, these contributions offer generalizable strategies for safe and efficient robotic operation:

- **Co-safe RL** for multi-agent systems that balances safety with task efficiency by incorporating neighbouring agents' safety models.
- **Model-free safety filtering** for soft robots that enables safe operation without relying on precise dynamic models.

These methods pave the way toward robust, adaptive, and safety-aware robotic systems capable of operating in complex, uncertain, and interactive environments.

Bibliography

[1] Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27): 68–73, 2015. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2015.11.154. Analysis and Design of Hybrid Systems ADHS. 2.5.2

[2] Joshua Achiam and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019. URL https://api.semanticscholar.org/CorpusID:208283920. 3.2

[3] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999. 3.2

[4] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019. doi: 10.23919/ECC.2019.8796030. 2.2

[5] Aaron D Ames, Xuan Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier functions: Theory and applications. *European Journal of Control*, 49:69–92, 2019. 1

[6] Yarden As, Ilnura Usmanova, Sebastian Curi, and Andreas Krause. Constrained policy optimization via bayesian world models, 2022. URL https://arxiv.org/abs/2201.09802. 3.2

[7] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances, 2017. URL https://arxiv.org/abs/1709.07523. 1, 2.2

[8] Luigi Berducci, Shuo Yang, Rahul Mangharam, and Radu Grosu. Learning adaptive safety for multi-agent systems, 2023. URL https://arxiv.org/abs/2309.10657. 2.4.2

[9] Agustin Castellano, Hancheng Min, Juan Andrés Bazerque, and Enrique Mallada. Learning safety critics via a non-contractive binary bellman operator, 2024. URL https://arxiv.org/abs/2401.12849. 3.2

[10] Hamza Chakraa, Edouard Leclercq, François Guérin, and Dimitri Lefebvre. A centralized task allocation algorithm for a multi-robot inspection mission with sensing specifications. *IEEE Access*, 11:99935–99949, 2023. doi: 10.1109/ACCESS.2023.3315130.

2.1

[11] Mo Chen and Claire J. Tomlin. Annual review of control , robotics , and autonomous systems hamilton – jacobi reachability : Some recent theoretical advances and applications in unmanned airspace management. 2019. URL https://api.semanticscholar.org/CorpusID:261659614. 2.2

[12] Richard Desatnik, Mikhail Khrenov, Zachary Manchester, Philip LeDuc, and Carmel Majidi. Optimal control for a shape memory alloy actuated soft digit using iterative learning control. In *2024 IEEE 7th International Conference on Soft Robotics (RoboSoft)*, pages 48–54, 2024. doi: 10.1109/RoboSoft60065.2024.10521965. 3.7.1

[13] Jingliang Duan, Zhengyu Liu, Shengbo Eben Li, Qi Sun, Zhenzhong Jia, and Bo Cheng. Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints. *Neurocomputing*, 484:128–141, May 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.04.134. URL http://dx.doi.org/10.1016/j.neucom.2021.04.134. 3.2

[14] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research*, 17(7):760–772, 1998. 2.2

[15] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 Int. Conf. on Robot. and Automat. (ICRA)*, pages 8550–8556. IEEE, 2019. 3.6.1.1

[16] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477. 3.6.1.2

[17] Zhan Gao, Guang Yang, and Amanda Prorok. Online control barrier functions for decentralized multi-agent navigation, 2023. URL https://arxiv.org/abs/2303.04313. 2.2

[18] Yang Guan, Yangang Ren, Qi Sun, Shengbo Eben Li, Haitong Ma, Jingliang Duan, Yifan Dai, and Bo Cheng. Integrated decision and control: Towards interpretable and computationally efficient driving intelligence, 2021. URL https://arxiv.org/abs/2103.10290. 3.2

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290. 3.5

[20] Homayoun Honari, Mehran Ghafarian Tamizi, and Homayoun Najjaran. Safety optimized reinforcement learning via multi-objective policy optimization, 2024. URL https://arxiv.org/abs/2402.15197. 3.2

[21] Jiechuan Jiang, Kefan Su, and Zongqing Lu. Fully decentralized cooperative multi-

agent reinforcement learning: A survey, 2024. URL https://arxiv.org/abs/2401.04934. 2.1

[22] Andrés C. Jiménez, Vicente García-Díaz, and Sandro Bolaños. A decentralized framework for multi-agent robotic systems. *Sensors*, 18(2), 2018. ISSN 1424-8220. doi: 10.3390/s18020417. URL https://www.mdpi.com/1424-8220/18/2/417. 2.1

[23] Yoshiaki Kuwata, Sertac Karaman, Ju Teo, Emilio Frazzoli, and Jonathan P How. Motion planning for urban driving using rrt with linear dynamics and linearized constraints. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3549–3556, 2011. 1

[24] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971. 3.6.1.2

[25] Yihao Liu, Xu Cao, Tingting Chen, Yankai Jiang, Junjie You, Minghua Wu, Xiaosong Wang, Mengling Feng, Yaochu Jin, and Jintai Chen. From screens to scenes: A survey of embodied ai in healthcare, 2025. URL https://arxiv.org/abs/2501.07468. 1

[26] Yiwei Lyu, Wenhao Luo, and John M Dolan. Probabilistic safety-assured adaptive merging control for autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10764–10770. Ieee, 2021. 2.2

[27] Yiwei Lyu, Wenhao Luo, and John M. Dolan. Adaptive safe merging control for heterogeneous autonomous vehicles using parametric control barrier functions. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 542–547, 2022. doi: 10.1109/IV51971.2022.9827329. 2.2, 2.4.1

[28] Ian M Mitchell and Claire J Tomlin. A comparison of methods for time-dependent reachability analysis. *Hybrid Systems: Computation and Control*, pages 153–168, 2007. 1

[29] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005. 1

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602. 3.5

[31] Quoc Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. *Proceedings of the 2016 American Control Conference (ACC)*, pages 322–328, 2016. 1

[32] Nilaksh, Abhishek Ranjan, Shreenabh Agrawal, Aayush Jain, Pushpak Jagtap, and Shishir Kolathaya. Barrier functions inspired reward shaping for reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, page 10807–10813. IEEE, May 2024. doi: 10.1109/icra57147.2024.10610391. URL http://dx.doi.org/10.1109/ICRA57147.2024.10610391. 2.4.2

[33] Hardik Parwana, Aquib Mustafa, and Dimitra Panagou. Trust-based rate-tunable control barrier functions for non-cooperative multi-agent systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2222–2229, 2022. doi: 10.1109/CDC51059.2022.9992744. 2.2

[34] Zach J. Patterson, Wei Xiao, Emily Sologuren, and Daniela Rus. Safe control for soft-rigid robots with self-contact using control barrier functions. In *2024 IEEE 7th International Conference on Soft Robotics (RoboSoft)*, pages 151–156, 2024. doi: 10.1109/RoboSoft60065.2024.10522000. 3.2

[35] Wilfrid Perruquetti and Jean-Pierre Barbot. A survey on the lyapunov based adaptive control and related results. *IMA Journal of Mathematical Control and Information*, 25(2):181–197, 2008. 1

[36] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Weighted buffered voronoi cells for distributed semi-cooperative behavior. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5611–5617, 2020. doi: 10.1109/ICRA40945.2020.9196686. 2.2

[37] James B Rawlings, David Q Mayne, and Moritz M Diehl. *Model predictive control: Theory, computation, and design*. Nob Hill Publishing, 2017. 1

[38] Andrew P. Sabelhaus, Zach J. Patterson, Anthony T. Wertz, and Carmel Majidi. Safe supervisory control of soft robot actuators. *Soft Robotics*, 11(4):561–572, 2024. doi: 10.1089/soro.2022.0131. URL https://doi.org/10.1089/soro.2022.0131. PMID: 38324015. 3.2

[39] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. doi: 10.1109/TASSP.1978.1163055. 1.

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347. 2.4.1, 3.6.1.2

[41] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic, 2020. URL https://arxiv.org/abs/2010.14603. 3.2

[42] Guo Ning Sue, Yogita Choudhary, Richard Desatnik, Carmel Majidi, John Dolan, and Guanya Shi. Q-learning-based model-free safety filter, 2024. URL https://arxiv.org/abs/2411.19809. 1, 3.4.2

[43] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html. 3.3.2

[44] Daniel CH Tan, Fernando Acero, Robert McCarthy, Dimitrios Kanoulas, and Zhibin Li. Value functions are control barrier functions: Verification of safe policies using control theory. 3.5.1

[45] Xiao Tan and Dimos V. Dimarogonas. Distributed implementation of control barrier functions for multi-agent systems. *IEEE Control Systems Letters*, 6: 1879–1884, 2022. URL https://api.semanticscholar.org/CorpusID:245029473. 2.2

[46] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization, 2018. URL https://arxiv.org/abs/1805.11074. 3.2

[47] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robot. and Automat. Letters*, 6(3):4915–4922, 2021. 3.2, 3.6.1.2

[48] Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future, 2022. URL https://arxiv.org/abs/2202.07789. 3.2, 3.4.1

[49] Ping Xuan and Victor Lesser. Multi-agent policies: from centralized ones to decentralized ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, AAMAS '02, page 1098–1105, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134800. doi: 10.1145/545056.545078. URL https://doi.org/10.1145/545056.545078. 2.1

[50] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. *CoRR*, abs/2010.03152, 2020. URL https://arxiv.org/abs/2010.03152. 3.2

[51] Moritz A. Zanger, Karam Daaboul, and J. Marius Zöllner. Safe continuous control with constrained model-based policy optimization, 2021. URL https://arxiv.org/abs/2104.06922. 3.2

[52] Yunpeng Zhai, Peixi Peng, Chen Su, and Yonghong Tian. Dynamic belief for decentralized multi-agent cooperative learning. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 344–352. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/39. URL https://doi.org/10.24963/ijcai.2023/39. Main Track. 2.1

[53] Jingyuan Zhao, Yuyan Wu, Rui Deng, Susu Xu, Jinpeng Gao, and Andrew Burke.

A survey of autonomous driving from a deep learning perspective. *ACM Comput. Surv.*, 57(10), May 2025. ISSN 0360-0300. doi: 10.1145/3729420. URL https://doi.org/10.1145/3729420. 1

[54] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017. doi: 10.1109/LRA.2017.2656241. 2.2