UNIVERSITÄT BONN

HUMANOID ROBOTS LAB

Predictive Safety in Reinforcement Learning

From MPC-Guidance to Learned Regulators

Dissertation

zur

Erlangung des Doktorgrades (*Dr. rer. nat.*)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Murad Elnagdi

aus

Port Said, Ägypten

Bonn, 2026

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachterin & Betreuerin:	Prof. Dr. Maren Bennewitz Rheinische Friedrich-Wilhelms-Universität Bonn
Gutachter:	Jun.-Prof. Dr. Aamir Ahmad University of Stuttgart

Tag der Promotion: 20.03.2026

Erscheinungsjahr: 2026

Acknowledgements

My deepest gratitude goes to my supervisor, Prof. Dr. Maren Bennewitz, for the opportunity to pursue my PhD in her lab and for the environment of intellectual freedom she created. Her steady guidance, clear advice, and patient feedback were invaluable, but it was the trust she placed in me to explore my own ideas that truly shaped my growth as a researcher. Thank you for helping me navigate this journey with confidence.

I owe a heartfelt debt to my father, Dawood Elnagdi. His constant encouragement and quiet strength have been my anchor, carrying me through the most difficult moments of this process. I am equally thankful to my brother, Khaled Elnagdi; he has been both my supporter and my honest critic. His unwavering belief in my abilities, even when I doubted them myself, has been a continuous source of motivation.

I am incredibly lucky to have friends who became family. Amr Bekhiet, Mohamad Hakam Shams Eddin, Mohamed Yaghmour, and Ahmed Shokry: thank you for the laughter, the perspective, and for keeping me sane during the long nights. Finally, a special thanks to Sicong Pan, Shahram Khorhishidi, Xuying Huang, and my lab colleagues. Whether it was a critical discussion or a thoughtful feedback, your generosity and brilliance made this work possible.

Abstract

Autonomous robots operating in unstructured environments require control policies that achieve their tasks reliably while respecting safety constraints. While Reinforcement Learning (RL) has emerged as a powerful data-driven paradigm for optimizing these policies through interaction, its application in robotics is hindered by three fundamental barriers during exploration: it is unstable under sparse rewards, since feedback signals are rare or delayed until the task is solved; it is unsafe, risking damage to the robot and its surroundings; and it is sample-inefficient, as it requires extensive interaction with the environment. This thesis addresses these challenges by establishing prediction as a framework for exploration. We combine model predictive control (MPC) and predicted safety signals to guide data collection, constrain risk, and preserve task performance during training and execution of RL agents. Our methodological progression begins by addressing the inefficiency of random exploration in sparse-reward settings. We introduce a hybrid training framework that utilizes MPC as a synthetic expert to guide the agent through complex navigation tasks. This approach accelerates convergence by alternating planned trajectories with trial-and-error experience, yielding a lightweight policy for independent deployment. Transitioning to the safety-critical domain of multi-robot systems, we subsequently employ predictive control as a distributed safety filter. We develop a scalable behavior-based formation controller secured by distributed nonlinear MPC shields, which ensures collision-free training, faster convergence, and enables zero-shot transfer to larger teams and to physical hardware. However, relying on static safety shields often induces conservative behavior that hinders learning. To overcome this limitation, we propose a dynamic safety shield that utilizes a supervisor agent to adapt constraint parameters online. By tuning the shield's sensitivity to the environment, we reduce solver failures and prevent the conservative behaviors typical of static shields. Ultimately, to eliminate the computational bottleneck of running optimization solvers at runtime, we transfer these predictive principles into a modular action regulator. This learned mechanism uses cost critics to preemptively adjust actions based on predicted risk, decoupling safety enforcement from reward maximization. Collectively, these studies show that combining short-horizon planning with learned risk estimation makes RL safer and more sample-efficient without sacrificing task performance. The proposed methods are evaluated in extensive simulation, with the MPC-based frameworks further validated on physical robots to demonstrate their robustness under real-world uncertainties.

Table of Contents

1 Introduction

1.1 Motivation

Autonomous robots are moving beyond structured factory floors into dynamic environments like hospitals [1], warehouses [2], agriculture [3], and planetary exploration [4]. Unlike in controlled industrial cells, these agents must plan, perceive, and act under significant uncertainty while satisfying safety constraints, specifically the requirement to interact with the world without damaging themselves or their surroundings. In such settings, autonomy goes beyond simple goal-reaching; it demands a balance between competing objectives, such as maximizing efficiency (e.g., speed and energy usage) while maintaining the conservative safety margins necessary to avoid obstacles. This creates a central challenge: transforming rich, high-dimensional sensory streams into real-time control actions that not only solve the task but also respect rigorous operational constraints, including physical actuator limits and the bounds of onboard computational capacity.

Classical pipelines [5], [6] decompose autonomy into perception, state estimation, mapping, motion planning, and control. This modular approach is well established and effective when models are accurate, the state is compact, and objectives are easily encoded. However, as tasks grow in complexity and involve high-dimensional sensory inputs, these stacks demand extensive feature engineering, requiring the manual design of intermediate state representations and cost functions that must be synchronized across modules. Furthermore, the interfaces between these isolated components can propagate and compound errors, often necessitating continual system-wide retuning. These limitations motivate a complementary learning-based route that seeks to automate feature extraction by mapping rich observations directly to actions, allowing the agent to adapt its behavior directly from interaction data.

Modern machine learning seeks to learn functions that map inputs to outputs directly from data, reducing manual feature extraction and enabling models to leverage high-dimensional sensing [7]. In supervised settings, models learn from labeled examples to predict actions or intermediate quantities; in robot learning from demonstration, this typically requires collecting extensive expert trajectories that cover the relevant state space and are of sufficient quality [8]. Such approaches excel when large, well-curated datasets reflect the deployment distribution and when the objective can be expressed as an immediate prediction error. However, for sequential decision making, the learner's predictions influence what it will observe next, goals are defined over long horizons, and feedback may arrive only after many steps. Purely supervised pipelines therefore struggle to close the loop. They depend on demonstrations and are vulnerable to distribution shift and error accumulation induced by the learner's own actions [8]. Furthermore, they offer no direct mechanism for long-horizon credit assignment—that is,

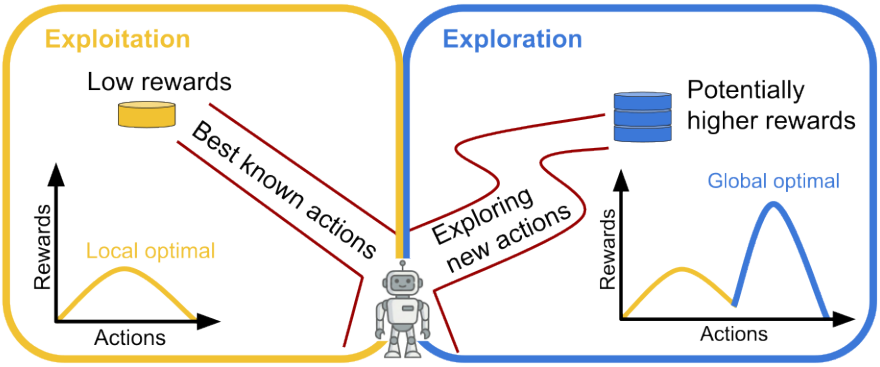


Figure 1.1: Illustration of the exploration–exploitation trade-off in reinforcement learning. Starting from the center, the robot can repeatedly exploit actions that keep it within the yellow region and lead to a nearby local optimum (left), or it can explore beyond this familiar region into the blue region (right), where a better optimum may exist. The red trajectories depict these alternative behaviors while the value curves at the bottom show how pure exploitation can get stuck in a local optimum, whereas exploration is needed to discover a better solution.

linking delayed outcomes back to the decisions that caused them [9].

Reinforcement learning (RL) addresses these issues by learning through interaction, rather than relying solely on fixed demonstrations. An agent observes the environment, selects actions, and updates its policy to maximize expected long-term return instead of one-step prediction accuracy [9]. This framing naturally captures delayed consequences, supports learning from trial-and-error experience, and in principle allows policies to map high-dimensional observations directly to actions. Beyond this conceptual appeal, RL has demonstrated strong results across sequential decision-making tasks—from human-level control in Atari [10] to grandmaster play in Go [11] and StarCraft [12]. In robotics, a central appeal of RL is that policies can be optimized end-to-end from sensory input to motor commands, reducing task-specific engineering of intermediate representations and control laws while adapting behavior from data [13]. Realizing this potential, however, requires the agent to autonomously discover behavior that yields high long-term reward while operating under the constraints and uncertainties of the physical world. This creates a fundamental tension: to maximize performance, the agent must rely on actions known to yield reward; yet to discover such actions in the first place, it must risk probing the unknown.

This dilemma brings into focus the central algorithmic question of the *exploration–exploitation trade-off* (see Fig. 1.1). Exploitation selects actions that are currently estimated to yield high return, turning existing knowledge into reward. Exploration, in contrast, deliberately visits actions and states whose outcomes are uncertain, in order to refine those estimates and discover better strategies. If exploration is too weak, the agent can prematurely commit to a suboptimal but familiar behavior and never gather evidence about superior alternatives. If exploitation is too weak, it may spend its time probing the environment without consolidating a high-performing policy, incurring unnecessary cost.

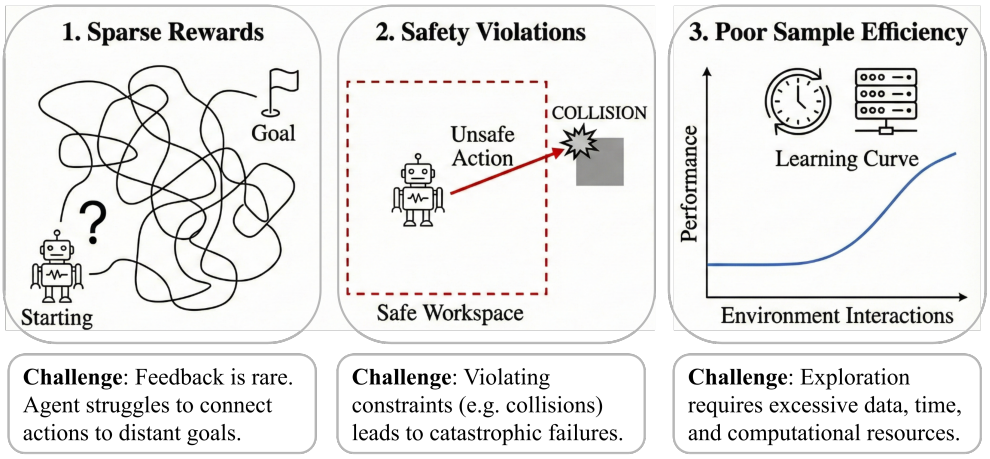


Figure 1.2: Visualizing the three central challenges of exploration in robotics addressed in this thesis. (**1**) In sparse reward settings, the lack of gradient signal leads to aimless wandering rather than purposeful learning. (**2**) Standard exploration mechanisms ignore physical constraints, leading to catastrophic failures such as collisions. (**3**) The trial-and-error nature of RL results in poor sample efficiency, making the learning process time- and resource-intensive.

Despite being essential for learning, exploration in robotics introduces persistent difficulties, see Fig. 1.2:

1. **Sparse rewards.** Many robotic manipulation and navigation tasks only provide reward when a goal is eventually achieved, and intermediate signals are weak or absent. Under such sparse rewards, random exploration rarely encounters success states, so the agent receives almost no gradient signal to improve its policy. Long horizons further exacerbate this problem by making credit assignment—linking delayed outcomes back to the decisions that caused them—highly non-trivial. As a result, learning becomes brittle and slow [14], [15].

2. **Safety.** Exploration inevitably drives the system into states it has not seen before, where estimated rewards are most inaccurate. In robotics, such states may correspond to collisions, unstable behaviors, or violations of hard constraints (for example, joint limits, torque bounds, or human safety zones). Such failures may be acceptable in simulation but are intolerable on physical systems, motivating the field of safe reinforcement learning and safe exploration [16], [17]. The core difficulty is trading off informative exploration against the requirement to remain within a safe operating envelope.

3. **Sample efficiency.** Exploration in RL is fundamentally a trial-and-error process: the agent must try actions, observe their consequences, and gradually adjust its policy. As a result, many deep RL algorithms require millions or billions of environment steps to discover effective policies in complex tasks [18]. This makes exploration extremely time- and resource-intensive and necessitates methods that can learn effectively from limited interaction rather than relying on prolonged online trial-and-error.

These challenges mean that exploration in robotics cannot be treated as a minor implementation detail. Instead, it must be designed to be efficient, safe, and compatible with the system's computational and operational constraints. In this thesis, these issues motivate augmenting RL with tools from optimal control, which can predict short-horizon system evolution and optimize actions under explicit costs and constraints, guiding exploration toward safe and informative experience.

Model Predictive Control (MPC) is the most prominent of these tools, offering a rigorous framework for acting under constraints. While originally pioneered in the process industries of the 1970s for regulating slow-moving chemical plants [19], [20], MPC has evolved far beyond its industrial origins. Historically, the heavy computational burden of solving optimization problems online limited its use in robotics applications which require fast online computation. However, the exponential growth in onboard computing power and the development of efficient numerical solvers have since propelled MPC into high-speed applications, ranging from autonomous racing [21], [22] to agile legged locomotion [23]. Unlike classical PID controllers [24] that merely react to instantaneous errors, MPC is anticipatory: it utilizes an internal model to predict future system evolution and solves a finite-horizon optimization problem at every control cycle. This allows it to handle multi-input multi-output (MIMO) systems naturally and, crucially, to enforce hard constraints on actuators and safety boundaries explicitly [25].

Conceptually, MPC and Reinforcement Learning share a fundamental resemblance: both seek to identify optimal actions by optimizing a cumulative objective function, framed as a reward maximization in RL or cost function minimization in MPC. Furthermore, both are inherently anticipatory, looking ahead to account for the future consequences of current decisions. However, they diverge in how they process information to achieve this goal. RL is a data-driven approach that distills trial-and-error experience into a policy, allowing it to approximate the optimal solution over a long or infinite horizon without needing to solve an optimization problem at runtime. MPC, in contrast, is model-driven: it relies on known physical equations to predict the immediate future and solves a finite-horizon optimization problem online at every time step. This rigorous online replanning makes MPC dependable for safety, but it introduces well-known limitations. First, solving nonlinear optimization problems at control frequency is computationally demanding [26]. Second, performance is highly sensitive to the accuracy of the dynamics model and the manual tuning of cost functions [27]. Third, standard formulations typically assume access to structured state representations and cannot directly ingest the raw, high-dimensional observations standard in modern learning pipelines [28]. Thus, neither RL nor MPC alone fully addresses the requirements for safe, efficient, and perceptually driven autonomy.

These observations motivate combining learning with model predictive control: use prediction to structure exploration rather than replace learning. This thesis therefore asks how MPC predictions and related predictive tools can serve as a framework for exploration; guiding data collection, constraining risk, and feeding useful signals into

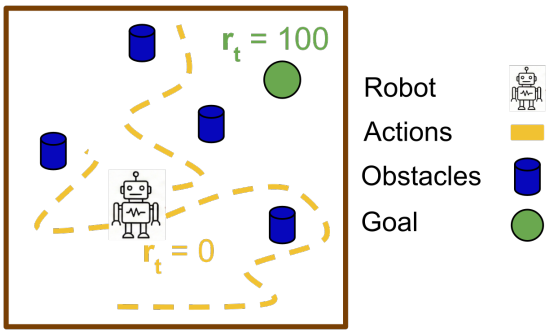


Figure 1.3: The sparse reward challenge in navigation. Since the environment provides no intermediate feedback ($r_t = 0$) and rewards the agent only upon success ($r_t = 100$), standard exploration devolves into an inefficient random walk (dashed line), often failing to discover the goal. **RQ1** addresses this limitation by investigating how MPC can be leveraged to guide the agent toward the target efficiently.

learning, while keeping the deployed policy lightweight and observation-driven.

1.2 Research Questions

Guided by this motivation, this thesis addresses four central research questions:

1.2.1 Accelerating Learning in Sparse Reward Settings

Reinforcement learning excels when feedback is dense and informative, but in many navigation and manipulation tasks, the agent receives a binary signal only upon success. Under such sparse rewards, standard exploration strategies often fail to discover the goal entirely because the agent cannot identify promising behaviors before reaching the target [14], [29]. While expert demonstrations can bootstrap learning, obtaining human data is costly and is often limited in quantity [30], [31]. Optimization-based controllers, such as MPC, can generate unlimited synthetic demonstrations, yet relying on them at runtime imposes a heavy computational burden, particularly for nonlinear systems with limited onboard compute [32], [33]. This raises the question of how to leverage the teacher during training only:

> **RQ1:** *How can MPC be utilized as an experience source to improve the training efficiency of an RL agent under sparse rewards, such that the final deployment remains a lightweight, policy-only controller without an MPC fallback?*

1.2.2 Safe and Scalable Multi-Robot Formation Control

While RQ1 addresses single-agent efficiency, real-world autonomy often requires coordinating teams. Safety in multi-agent reinforcement learning (MARL) has typically been addressed in scenarios where robots pursue independent, individual goals [34], [35]. In such settings, safety can often be achieved simply by extending pre-trained

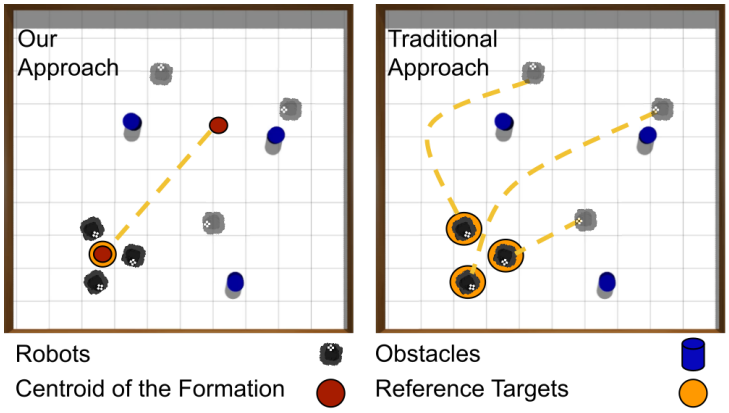


Figure 1.4: Comparison of the proposed behavior-based formulation against traditional approaches. **Right (Traditional):** Requires assigning individual reference targets and path plans to every robot, which limits scalability and demands complex pre-planning with increasing number of robots. **Left (Ours):** We introduce a scalable formulation where the team tracks a single collective target for the formation's centroid. This eliminates the need for individual path planners but forces agents to navigate in constant, close proximity to maintain relative formation, creating the distinct safety challenges addressed in RQ2.

single-agent policies to multiple actors [36], as agents require only intermittent collision avoidance. However, these standard approaches necessitate a reference target for each individual robot. This requirement introduces significant computational complexity by demanding a dedicated path planner for every agent, acting as a bottleneck for scalability as team size grows. To eliminate this bottleneck, we introduce a behavior-based formulation that relies on a single collective objective: steering the formation's centroid, see Fig. 1.4. While this approach enables scalability by removing the burden of individual planning, it creates a significantly more challenging landscape. Unlike independent navigation, this centroid-based formulation compels agents to navigate in constant, close proximity to maintain relative formation, drastically increasing the frequency and complexity of potential collisions. Consequently, the challenge lies in developing a framework that maintains this scalability without compromising physical safety. This motivates our second research question:

> **RQ2:** *How can behavior-based cooperative navigation be formulated to enable scalable formation control without individual reference targets, while ensuring collision-free training and execution in multi-robot scenarios?*

1.2.3 Adapting Safety Shields to Preserve Exploration

While the distributed MPC filters utilized in RQ2 effectively prevent collisions, they rely on static parameters and hard constraints manually tuned prior to deployment [37]. Such rigid, pre-tuned shields often face a dilemma (see Fig. 1.5): they either overly restrict the task agent's exploration, resulting in suboptimal rewards, or suffer from "solver failure" where the MPC cannot find a solution satisfying all hard constraints in complex environments, causing the agent to get stuck [34], [38]. To mitigate this, the

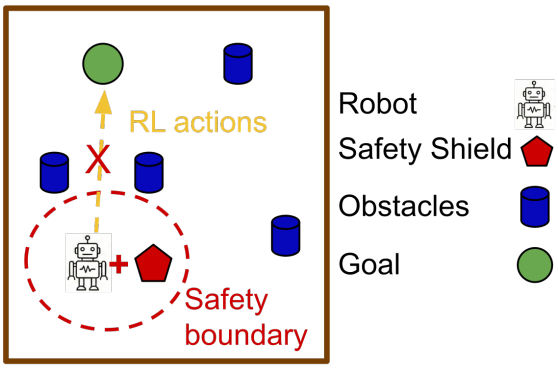


Figure 1.5: The dilemma of static safety shields. The RL agent proposes a path through the obstacles (yellow arrow), but the pre-tuned safety boundary (red dashed circle) is too wide to fit through the gap. This results in a "solver failure" (red X) where the shield blocks the action and the robot freezes, even though a feasible path exists. **RQ3** addresses this by dynamically adapting the shield parameters in real-time.

shield must be adaptive rather than fixed; capable of adjusting weights online to balance safety with the agent's need to explore. This necessitates a mechanism to tune the safety-shield in real-time without human intervention:

> **RQ3:** *How can an MPC-based safety shield be dynamically tuned via a secondary supervisor agent to minimize solver failures and collisions, without overly constraining the primary RL agent's ability to explore?*

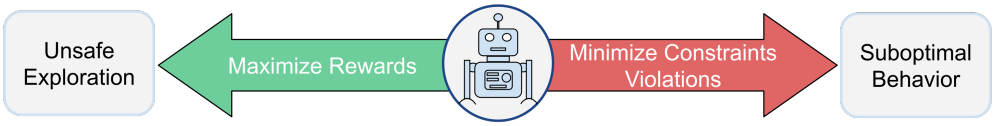


Figure 1.6: Visualizing the objective conflict in Constrained RL. The agent is pulled between two competing goals: maximizing rewards, which encourages unsafe exploration (left), and minimizing constraint violations, which often leads to over-conservative or suboptimal behavior (right). **RQ4** addresses this by decoupling these objectives using a learned regulator.

1.2.4 Efficient Safety via Learned Action Regulation

Finally, while MPC shields provide rigorous safety guarantees, they depend on solving numerical optimization problems at every control cycle, which requires significant onboard compute and prior system knowledge. Constrained RL, as illustrated in Fig. 1.6, attempts to embed safety into the policy but often suffers from training instability due to conflicting objectives (reward maximization vs. cost minimization) [39], [40]. A middle ground is required: a method that retains the "look-ahead" safety of MPC but enables fast execution without requiring prior knowledge of the system. This motivates our final research question on replacing explicit optimization with learned regulation:

RQ4: *How can predicted constraint violations be leveraged to modulate actions via a learned regulator, preserving exploration while eliminating the need for an online solver?*

1.3 Contributions

This thesis establishes a unified framework for safe and efficient robot learning by integrating the rigorous foresight of control theory with the adaptive capabilities of reinforcement learning. We demonstrate that predictive signals—whether derived from explicit MPC solvers or learned neural critics—can structure exploration, enforce safety, and guide data collection, all while ensuring the final deployed policy remains lightweight and observation-driven. The core contributions are structured as follows:

1.3.1 Accelerating Exploration via MPC Demonstrations

In chapter 3, we address the inefficiency of random exploration in sparse-reward settings by introducing a framework that utilizes MPC as an *experience source*. Rather than relying on costly human demonstrations, we interleave MPC-generated successful trajectories with the agent's trial-and-error experience. We demonstrate that this hybrid approach effectively bridges the gap between initial conditions and sparse goals, significantly improving convergence rates and success in mobile robot navigation compared to pure RL baselines. Crucially, the MPC is used strictly for training guidance, leaving a fast, standalone policy for deployment.

1.3.2 Distributed Safety Filters for Behavior-Based Multi-Robot Navigation

Moving to multi-agent systems, in chapter 4 we propose a novel formulation for behavior-based cooperative navigation that relies on a single formation centroid rather than individual reference targets, enabling scalability to larger teams without complex pre-planning. To secure the safety of this high-interaction setup, we integrate distributed MPC safety filters that ensure zero collisions during both training and execution. We validate this approach on real robots, demonstrating that behavior-based formations can be learned safely and efficiently even under the non-stationary dynamics of multi-agent interaction.

1.3.3 Adaptive Shielding via Hierarchical Supervision

Recognizing that static safety shields can be overly conservative or prone to solver failure, in chapter 5 we develop a *Dynamic Safety Shield* tuned by a secondary supervisor agent. This supervisor learns to adjust the shield's weights online, aligning the safe recovery actions with the task agent's objectives. This contribution solves the "freezing robot" problem common in hard-constrained shielding: by adapting constraints to the

immediate environmental complexity, we minimize solver failures and allow for aggressive yet safe exploration, improving the trade-off between goal reaching and collision avoidance.

1.3.4 Efficient Safety via Constraint-Aware Action Regulation

Finally, to eliminate the computational burden of running an optimization solver at runtime, in chapter 6 we propose a modular *Action Regulator*. This mechanism distills the principles of predictive shielding into a learned neural network that scales actions based on future violation probabilities predicted by twin cost critics. This approach decouples reward maximization from safety enforcement, avoiding the instability of joint optimization while preserving exploration. We show that this method achieves state-of-the-art performance ratios on safety benchmarks, providing the safety benefits of look-ahead control with the inference speed of a model-free policy.

1.4 Publications Included in the Thesis

The core results of this thesis have appeared as the following publications; the chapters map to the corresponding articles.

- **Chapter 3:** M. Dawood, N. Dengler, J. de Heuvel, and M. Bennewitz. *Handling Sparse Rewards in Reinforcement Learning Using Model Predictive Control*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023. DOI: https://doi.org/10.1109/ICRA48891.2023.10161492
- **Chapter 4:** M. Dawood, S. Pan, N. Dengler, S. Zhou, A. P. Schöllig, and M. Bennewitz. *Safe Multi-Agent Reinforcement Learning for Behavior-Based Cooperative Navigation*. *IEEE Robotics and Automation Letters (RA-L)*, 2025. DOI: https://doi.org/10.1109/LRA.2025.3560830
- **Chapter 5:** M. Dawood, A. Shokry, and M. Bennewitz. *A Dynamic Safety Shield for Safe and Efficient Reinforcement Learning of Navigation Tasks*. In *Proceedings of the 7th Annual Learning for Dynamics & Control Conference (L4DC)*, 2025. DOI: https://doi.org/10.48550/arXiv.2412.04153
- **Chapter 6:** M. Dawood, U. A. Siddiquie, S. Khorshidi, and M. Bennewitz. *Constraint-Aware Reinforcement Learning via Adaptive Action Scaling*. In *Proceedings of the 8th Annual Learning for Dynamics & Control Conference (L4DC)*, 2026. DOI: https://doi.org/10.48550/arXiv.2510.11491

1.5 Publications Not Covered by This Thesis

The following publications were written during my appointment as a research associate but are not included in this dissertation:

- S. Khorshidi*, M. Dawood*, and M. Bennewitz. "Centroidal State Estimation Based on the Koopman Embedding for Dynamic Legged Locomotion." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. (equal contribution)
- S. Khorshidi, M. Dawood, B. Nederkorn, M. Bennewitz, and M. Khadiv. "Physically-Consistent Parameter Identification of Robots in Contact." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- A. Shokry, W. Gomaa, T. Zaenker, M. Dawood, S. A. Maged, M. I. Awad, and M. Bennewitz. "Context-Based Meta Reinforcement Learning for Robust and Adaptable Peg-in-Hole Assembly Tasks." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

1.6 Collaborations

Because portions of this work were conducted jointly with other researchers, the thesis adopts the plural "we." The following summary clarifies collaboration scope and my contributions for each chapter.

- **Chapter 3:** Co-developed with N. Dengler and J. de Heuvel; both contributed discussions and ideas during the development of the work.
- **Chapter 4:** Co-authored in collaboration with S. Pan, N. Dengler, S. Zhou, and A. P. Schoellig. S. Pan advised the design of the experimental evaluation; N. Dengler, S. Zhou, and A. P. Schoellig contributed technical discussions and ideas during the development of the work.
- **Chapter 5:** This work was co-authored with A. Shokry. A. Shokry implemented and ran the experiments for two of the baselines.
- **Chapter 6:** Conducted together with U. A. Siddiquie and S. Khorshidi. U. A. Siddiquie implemented the baselines; S. Khorshidi provided technical insights and fruitful discussions.

2 Background

This chapter collects the technical background used throughout the thesis. We first introduce the Markov Decision Process formalism for reinforcement learning (RL), including policies, value functions, and the Bellman relations. We then present the learning objective and the actor–critic architecture used in our implementations. Next, we review model predictive control (MPC) for constrained optimal control with emphasis on prediction models and optimal control problems.

2.1 Reinforcement Learning Background

Reinforcement Learning (RL) formalizes the problem of goal-directed learning from interaction. Unlike supervised learning, which relies on a fixed dataset of labeled examples, an RL agent learns by actively exploring its environment, observing the consequences of its actions, and optimizing its behavior to maximize a cumulative reward signal [9]. This trial-and-error process is illustrated in Fig. 2.1: at each discrete time step t, the agent receives an observation o_t (representing the state s_t), executes an action a_t, and receives a scalar reward r_t along with the next observation o_{t+1}.

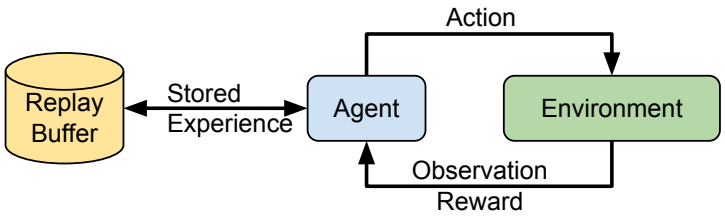
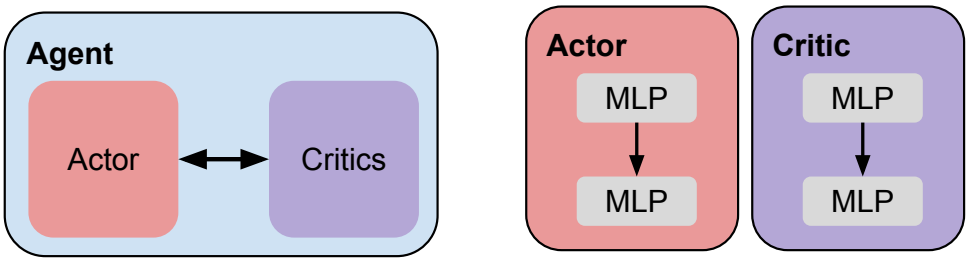


Figure 2.1: Reinforcement learning overview. The agent selects an action a_t based on the current state s_t; the environment transitions to s_{t+1} and returns a reward r_t. These transitions are stored in a replay buffer $\mathcal{D}$ for off-policy training.

2.1.1 Markov Decision Processes (MDP)

We model this interaction mathematically as a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$. The state space $\mathcal{S} \subseteq \mathbb{R}^n$ and action space $\mathcal{A} \subseteq \mathbb{R}^d$ represent the continuous dimensions of the robot and its actuators. The dynamics are governed by the transition distribution $P(s' \mid s, a)$, which gives the probability of moving to state s' given the current state and action. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines the immediate feedback, and the discount factor $\gamma \in (0, 1)$ determines the importance of future rewards.

The agent's goal is to learn a policy π, which maps states to actions (or distributions



(a) Agent architecture (b) Network components

Figure 2.2: Actor–Critic Architecture. (a) The agent consists of an actor that interacts with the environment and a critic that evaluates performance. (b) Both are parameterized as deep neural networks mapping high-dimensional observations to actions and values, respectively.

over actions), to maximize the expected discounted return $J(\pi)$:

$$J(\pi) \;=\; \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\, r(s_t,a_t)\right], \tag{2.1}$$

where the expectation is taken over trajectories τ induced by the policy and system dynamics.

2.1.2 Actor–Critic Architecture

To solve continuous control tasks, we utilize an Actor–Critic architecture (Fig. 2.2a). This approach maintains two distinct function approximators (neural networks):

1. **The Actor (π_θ):** Learns to select actions that maximize the value estimated by the critic. For continuous action spaces, the policy is often parameterized as a deterministic mapping $\mu_\theta(s)$ or a squashed Gaussian distribution.

2. **The Critic (Q_ϕ):** Learns to estimate the expected return of taking action a in state s and following the policy thereafter.

The value of a state-action pair under policy π is formally defined as the action-value function Q^π:

$$Q^\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\Big|\, s_0=s,\; a_0=a\Big]. \tag{2.2}$$

This function satisfies the recursive Bellman equation, which serves as the foundation for training the critic:

$$Q^\pi(s,a) = r(s,a) + \gamma\,\mathbb{E}_{s'\sim P}\big[\,Q^\pi(s',\pi(s'))\,\big]. \tag{2.3}$$

The actor is then updated by following the gradient of the critic's estimate with respect to the policy parameters θ:

$$\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_\theta\pi_\theta(s)\nabla_a Q_\phi(s,a)|_{a=\pi_\theta(s)}\right]. \tag{2.4}$$

2.1.3 Learning Stability and Efficiency

Training deep neural networks with RL introduces challenges not present in supervised learning, specifically data correlation and non-stationary targets. We address these using two standard mechanisms:

2.1.3.1 Replay Buffers

To break the temporal correlation of sequential data, transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer $\mathcal{D}$. During training, we sample mini-batches uniformly from $\mathcal{D}$. This allows the agent to learn from past experiences repeatedly and stabilizes the stochastic gradient descent updates [41].

2.1.3.2 Target Networks and TD Learning

The critic is trained via Temporal Difference (TD) learning to minimize the Bellman error. However, because the target value depends on the network's own parameters, optimization can diverge. To mitigate this, we employ a target network $Q_{\bar{\phi}}$, which is a slowly updating copy of the main critic. The target parameters $\bar{\phi}$ track the main parameters ϕ via Polyak averaging (soft updates):

$$\bar{\phi} \leftarrow \tau\,\phi + (1-\tau)\,\bar{\phi}, \quad \tau \ll 1. \tag{2.5}$$

This creates a stable objective for the critic update, significantly improving convergence in continuous control tasks [42].

2.2 Deep Neural Networks for Function Approximation

While Reinforcement Learning provides the mathematical framework for decision making (Sec. 2.1), the complexity of robotic tasks requires powerful function approximators to represent policies π_θ and value functions Q_ϕ. Historically, RL relied on linear function approximators or look-up tables, which required extensive domain expertise to engineer state features manually [9]. The advent of Deep Learning (DL) shifted this paradigm, allowing agents to learn end-to-end representations directly from raw sensory data, such as high-dimensional LiDAR scans, images, and proprioceptive signals like joint positions and velocities [43].

The theoretical foundation for this capability is the *Universal Approximation Theorem*, which states that a feedforward network with a single hidden layer and sufficient width can approximate any continuous function to arbitrary precision [44]. In this thesis, we leverage three specific neural architectures tailored to different modalities of robotic data.

2.2.1 Multilayer Perceptrons (MLPs)

The Multilayer Perceptron (MLP) serves as the backbone for processing low-dimensional, structured state information, such as robot proprioception (positions, velocities) or relative goal coordinates. An MLP consists of a sequence of fully connected layers. Mathematically, the output $h^{(l)}$ of layer l is computed from the input $h^{(l-1)}$ as:

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)}), \tag{2.6}$$

where W and b are learnable weights and biases, and $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU or Tanh). In the context of RL, MLPs are typically used as the final "decision heads" of the actor and critic networks, fusing features extracted by more complex encoders to output motor commands or value estimates.

2.2.2 Convolutional Neural Networks (CNNs) for Perception

When dealing with spatially structured sensory data, such as camera images or LiDAR occupancy grids, MLPs become computationally prohibitive due to the massive number of parameters required. Convolutional Neural Networks (CNNs) address this by enforcing local connectivity and weight sharing, allowing the network to detect translation-invariant features [45].

In robotic navigation, we specifically utilize CNNs to process LiDAR data. While raw LiDAR streams are often treated as 1D vectors, they represent spatial geometry. A 1D-CNN slides a kernel over the range data to extract local geometric features (e.g., corners, walls), while 2D-CNNs can process local occupancy grids constructed from the scan history. This allows the agent to build a "mental map" of obstacles directly from sensor streams, reducing the reliance on external mapping stacks [10].

2.2.3 Transformers and Multi-Head Attention

While MLPs and CNNs excel at processing fixed-size vectors or grid-structured data, they lack a native mechanism to reason about sets of objects or long temporal sequences where the relationship between elements is crucial. To address this, we leverage the transformer architecture [46], which utilizes *Self-Attention* to dynamically model dependencies between input tokens.

2.2.3.1 Self-Attention Mechanism

The core of the Transformer is the Scaled Dot-Product Attention. Fundamentally, this mechanism allows the network to determine how much focus (or "attention") a specific token should place on every other token in the input sequence. For each input element, the network computes three distinct vectors via linear projections: a *Query* q_i, a *Key* k_i, and a *Value* v_i.

- **Query** (Q)**:** Represents the feature the current token is looking for in others.

- **Key (K):** Represents the identifying feature of a token that queries match against.
- **Value (V):** Contains the actual information content to be extracted.

The attention weights are computed by taking the dot product between the Query of one token and the Keys of all others. A higher dot product implies a stronger relationship (or similarity) between the tokens. These scores are scaled and normalized via a softmax function to produce a probability distribution:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.7}$$

where d_k is the dimension of the key vectors. The scaling factor $\frac{1}{\sqrt{d_k}}$ prevents the dot products from growing too large in magnitude, which would otherwise push gradients into regions with vanishing derivatives.

2.2.3.2 Multi-Head Attention

A single attention pass typically captures one type of relationship (e.g., temporal proximity or semantic similarity). However, complex tasks often require tracking multiple distinct types of relations simultaneously. Multi-head attention addresses this by projecting the queries, keys, and values into h different subspaces and computing attention independently in parallel.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.8}$$

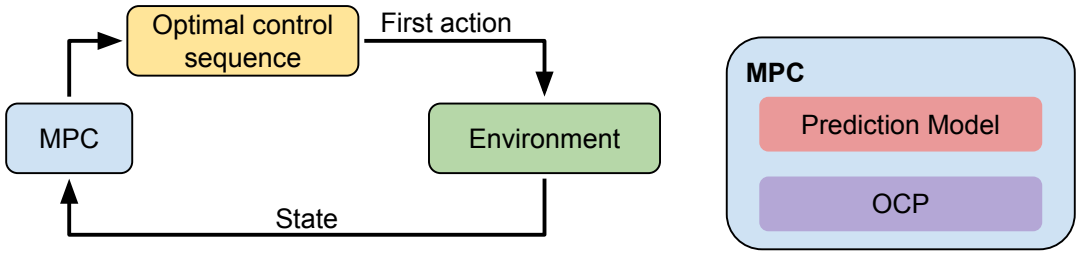
where each $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. This allows the model to attend to information from different representation subspaces at different positions effectively, enabling the network to build a rich, context-aware representation of the global state.

2.2.4 Stochastic Gradient Optimization

Training the neural architectures described above requires finding the optimal parameters θ that minimize a task-specific loss function $\mathcal{L}(\theta)$, such as the Bellman error for critics or the negated expected return for actors. Gradients of this loss with respect to the parameters are computed via the backpropagation algorithm and used to update the network weights iteratively.

However, optimization in reinforcement learning presents unique challenges compared to supervised learning: the data distribution is non-stationary because it shifts as the policy improves, and the gradient estimates are highly stochastic due to the sampling of actions and environment transitions. Standard stochastic gradient descent (SGD) often struggles in these settings due to its sensitivity to learning rate selection and its inability to handle varying gradient scales across different parameters.

To address this, we employ Adam (Adaptive Moment Estimation) [47]. Unlike standard SGD, Adam computes individual adaptive learning rates for different parameters.



(a) MPC Overview. (b) MPC components.

Figure 2.3: MPC overview and structure. (a) At each time step, the MPC outputs the optimal future control sequence, but only the first action is executed. After executing the action, the MPC receives an updated state and recalculates a new control sequence (receding horizon). (b) The MPC formulation relies on the interaction between a prediction model and an Optimal Control Problem (OCP), the latter of which encapsulates the objective function and constraints.

Conceptually, it achieves this by maintaining exponential moving averages of the gradients (momentum) and the squared gradients (variance) over time. By scaling the parameter updates based on these moving averages, the optimizer effectively normalizes the step size according to the confidence in the gradient direction. This adaptive scaling makes the optimization significantly more robust to the noisy, high-variance signals characteristic of deep reinforcement learning.

2.3 Model Predictive Control Fundamentals

2.3.1 Historical Context and the Receding Horizon Principle

Model Predictive Control (MPC) refers to a class of control algorithms that utilize an explicit process model to predict the future behavior of a system and optimize a control sequence over a finite time horizon [48]. Originally pioneered in the process industries of the 1970s for regulating slow-moving chemical plants [19], MPC has evolved into a standard tool for high-speed robotics due to advances in embedded optimization.

The core operation of MPC follows the *Receding Horizon* principle, illustrated in Fig. 2.3a. At each discrete time step t, the controller:

1. **Predicts** the system evolution over a horizon N using an internal model.
2. **Optimizes** a sequence of actions to minimize a cost function subject to constraints.
3. **Executes** only the first action a_t on the physical system.
4. **Shifts** the horizon to $t + 1$, updates the state estimate, and repeats the process.

This iterative replanning provides robust feedback, allowing the controller to correct for model mismatches and external disturbances naturally.

2.3.2 Prediction Model

While the previous section utilized s to denote the state in the context of reinforcement learning, here we adopt the control-theoretic convention of using x. The foundation of the MPC formulation (Fig. 2.3b) is the prediction model. Let $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and $a \in \mathcal{A} \subseteq \mathbb{R}^d$ denote the state and action vectors, respectively. The discrete-time dynamics are given by:

$$x_{t+1} \;=\; f(x_t, a_t), \qquad f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}. \tag{2.9}$$

Given the current state x_t, the controller unrolls these dynamics to form predicted states $x_{t+j|t}$ for $j = 0, \ldots, N$ under a candidate action sequence.

The fidelity of f represents a critical design trade-off. High-capacity nonlinear models (e.g., full rigid-body dynamics) capture inertial effects accurately but result in expensive, non-convex optimization problems. Simpler models (e.g., kinematic approximations) reduce solve times but introduce model mismatch. The choice of f and the horizon N must therefore balance physical accuracy against the real-time runtime constraints of the target hardware.

2.3.3 The Optimal Control Problem

Model Predictive Control relies on solving an optimization problem over a finite future horizon T. In its most general form, this is formulated as a continuous-time Optimal Control Problem (OCP). Let $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ denote the state and $a(t) \in \mathcal{A} \subseteq \mathbb{R}^d$ the control input at time t. The objective is to find the optimal trajectories $x(\cdot)$ and $a(\cdot)$ over the interval $t \in [0, T]$ that minimize a cost functional:

$$\min_{x(\cdot), a(\cdot)} \quad \int_0^T \ell(x(t), a(t))\, dt \;+\; V_f(x(T)) \tag{2.10}$$

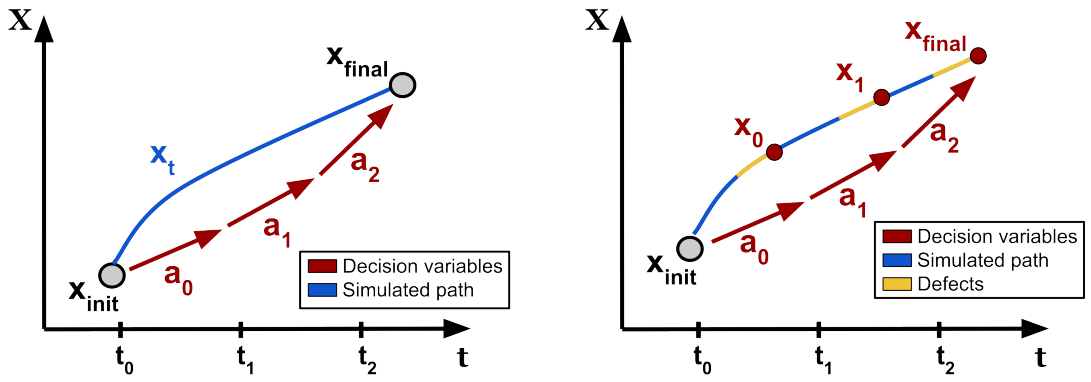
$$\begin{aligned}
\text{s.t.} \quad & \dot{x}(t) = f_c(x(t), a(t)), \quad t \in [0, T], && \text{(Dynamics)} \\
& g(x(t), a(t)) \leq 0, \quad t \in [0, T], && \text{(Path Constraints)} \\
& x(0) = x_{\text{init}}. && \text{(Initial State)}
\end{aligned}$$

Here, f_c represents the continuous-time system dynamics, $\ell(\cdot)$ is the stage cost, and $V_f(\cdot)$ is the terminal cost. In standard tracking MPC, these objectives are typically parameterized as quadratic functions to penalize deviations from a reference trajectory x_{ref}:

$$\ell(x, a) = \|x - x_{\text{ref}}\|_Q^2 + \|a\|_R^2, \tag{2.11}$$

$$V_f(x) = \|x - x_{\text{ref}}\|_Q^2, \tag{2.12}$$

where $\|x\|_W^2 = x^T W x$ denotes the weighted squared Euclidean norm. The matrices Q and R are the user-tuned state and control weighting matrices, respectively. Typically, Q is chosen to be diagonal positive semi-definite (penalizing tracking error), while R is



(a) Single Shooting

(b) Multiple Shooting

Figure 2.4: Visual comparison of transcription methods. **(a) Single Shooting:** The state trajectory (blue line) is fully determined by integrating dynamics from the fixed initial state x_{init} using the control sequence a_k. The optimization variables are limited solely to the controls. **(b) Multiple Shooting:** The problem is lifted to include intermediate state nodes x_k (red dots) as independent decision variables. The dynamics are integrated independently over each interval (blue lines), creating discontinuities (defects) between the integration result and the next decision variable node. The solver minimizes these defects to enforce continuity.

positive definite (penalizing control effort).

2.3.4 Direct Transcription to Nonlinear Programming

Since Eq. 2.10 is an infinite-dimensional problem, it cannot be solved directly by digital computers. To proceed, we employ *direct transcription methods* [49] to discretize the continuous problem into a finite-dimensional Nonlinear Programming (NLP) problem. We first partition the time horizon T into N intervals of length $T_s = T/N$. The continuous controls are typically parameterized as piecewise constant (zero-order hold), $a(t) = a_k$ for $t \in [t_k, t_{k+1})$. How the state trajectory $x(t)$ is handled determines the structure of the resulting NLP. Two primary transcription strategies are used: Single Shooting and Multiple Shooting.

2.3.4.1 Sequential Approach (Single Shooting)

In the Single Shooting approach, the optimization and simulation steps are strictly sequential. The optimizer treats the state trajectory as a dependent function of the controls and the initial state, denoted as $x_k(a_0, \dots, a_{k-1}, x_{\text{init}})$. This implicit trajectory is obtained by numerically integrating the system dynamics forward from the initial condition, as illustrated in Fig. 2.4a. Let $f_d(x_k, a_k)$ denote the discretized dynamics function obtained by integrating the continuous ODE over one sampling interval T_s. The resulting NLP optimizes only the discrete control actions $\{a_0, \dots, a_{N-1}\}$:

$$\begin{aligned} \min_{a_0,\dots,a_{N-1}} \quad & \sum_{k=0}^{N-1} \ell(x_k, a_k) + V_f(x_N) && (2.13)\\ \text{s.t.} \quad & x_{k+1} = f_d(x_k, a_k), \quad k = 0, \dots, N-1, && \text{(Dynamics)}\\ & g(x_k, a_k) \leq 0, \quad k = 0, \dots, N-1. && \text{(Path Constraints)}\\ & x_0 = x_{\text{init}}, && \text{(Initial State)} \end{aligned}$$

While this reduction in variables creates a compact problem, it imposes a strict computational dependency: if the solver updates an early control action, the entire subsequent trajectory must be re-simulated via f_d (discretized dynamics). This dependence leads to *poor numerical conditioning*, as small perturbations in the initial control actions propagate and amplify through the integration horizon, causing large deviations in the terminal state. This high sensitivity makes the optimization landscape difficult to traverse for unstable or highly nonlinear systems [50]. Furthermore, because the intermediate states are not included as decision variables, the solver cannot leverage prior knowledge to initialize the state trajectory directly, limiting the ability to warm-start the solver in complex environments.

2.3.4.2 Simultaneous Approach (Multiple Shooting)

In contrast to the sequential approach, Multiple Shooting lifts the problem dimension by treating both the control inputs and the intermediate state vectors at each shooting node as independent decision variables. Instead of integrating the dynamics over the entire horizon, the system dynamics are enforced locally as equality constraints. These constraints ensure that the **defects**—the gaps between the integrated state $f_d(x_k, a_k)$ and the next decision variable x_{k+1}—are driven to zero (see Fig. 2.4b). The resulting NLP optimizes over the joint set of states and controls:

$$\begin{aligned} \min_{\substack{x_0,\dots,x_N \\ a_0,\dots,a_{N-1}}} \quad & \sum_{k=0}^{N-1} \ell(x_k, a_k) + V_f(x_N) && (2.14)\\ \text{s.t.} \quad & x_{k+1} = f_d(x_k, a_k), \quad k = 0, \dots, N-1, && \text{(Dynamics (Defects))}\\ & g(x_k, a_k) \leq 0, \quad k = 0, \dots, N-1, && \text{(Path Constraints)}\\ & x_0 = x_{\text{init}}, && \text{(Initial State)} \end{aligned}$$

This simultaneous treatment offers two decisive advantages. First, it isolates the state at each time step, preventing numerical errors from accumulating and propagating over the horizon; this ensures stability even for highly nonlinear systems. Second, it allows the solver to be *warm-started* with a guess for the entire trajectory. By initializing the solver close to the solution using the previous time step's result, the computational time required to converge is drastically reduced, enabling high-frequency control [51]. Con-

sequently, the multiple shooting method serves as the foundational transcription strategy for the controllers developed in this work.

2.3.5 MPC Design Parameters

The efficacy of the NMPC controller relies on the careful tuning of three hyperparameters: the sampling time (T_s), the prediction horizon (N), and the weighting matrices (Q, R). These parameters dictate the trade-off between closed-loop performance (stability, tracking accuracy) and computational feasibility.

2.3.5.1 Sampling Time and Prediction Horizon

The sampling time T_s defines the discretization grid and the frequency at which the OCP is solved. Its selection is a compromise between capturing fast system dynamics and adhering to the hardware's computational budget:

- **Small T_s (High Frequency):** Necessary for agile platforms to reject disturbances and prevent aliasing. However, for a given number of shooting nodes N, a smaller T_s reduces the total physical look-ahead $(N \times T_s)$. Maintaining a sufficient horizon therefore requires a larger N, which increases the number of decision variables and significantly elevates the computational cost.
- **Large T_s (Low Frequency):** Reduces the computational burden but increases the discretization error. If T_s is too large compared to the system's time constants, the controller may react too slowly to changes, leading to instability.

Once T_s is established, the integer horizon length N determines the physical look-ahead time $(T_{hor} = N \times T_s)$. This introduces a critical safety trade-off. A short horizon results in **short-sightedness**, preventing the robot from anticipating distant constraints (e.g., braking early for a wall) and leading to inevitable collisions. Conversely, a long horizon ensures recursive feasibility and stability but increases the size of the optimization problem. In practice, N must be chosen such that the look-ahead time T_{hor} is sufficient for the system to reach a safe terminal state or satisfy stability constraints [25]. For example, in navigation tasks, the horizon should ideally exceed the robot's stopping distance to ensure that a safe braking trajectory always exists within the solver's look-ahead.

2.3.5.2 Cost Function Weights

The weighting matrices Q and R (introduced in Eq. Equation (2.11)) define the controller's priorities. Since state and control variables often have different physical units (e.g., meters vs. radians), these matrices also serve as normalizers.

- **State Weights (Q):** A diagonal matrix penalizing tracking error. Setting large values for specific diagonal elements forces the controller to track those states tightly.

However, excessively high penalties force the solver to correct even minute deviations immediately. This causes the controller to overreact to sensor noise, resulting in jittery or oscillatory motion.

- **Control Weights** (R)**:** A diagonal matrix penalizing control effort. Increasing elements of R acts as a "move suppression" factor, penalizing rapid changes in actuation. This yields smoother, energy-efficient trajectories but results in a more sluggish transient response.

This chapter has established the pillars of our technical approach: reinforcement learning provides the framework for autonomous discovery, and model predictive control ensures constraint-aware planning. While these fields are traditionally treated as distinct, the following chapter explores their synergy. Specifically, we address a core limitation of RL—the "reward shaping" bottleneck—by utilizing the predictive capabilities of MPC to guide agents through environments where feedback is sparse.

3 Handling Sparse Rewards in Reinforcement Learning Using Model Predictive Control

Abstract

The theoretical frameworks of reinforcement learning and model predictive control established in the previous chapter provide the necessary tools for autonomous robotic control. The work presented in this chapter investigates how these two methodologies can be integrated to facilitate learning in environments where feedback is naturally limited. Reinforcement learning (RL) has recently proven great success in various domains. Yet, the design of the reward function requires detailed domain expertise and tedious fine-tuning to ensure that agents are able to learn the desired behaviour. Using a sparse reward conveniently mitigates these challenges. However, the sparse reward represents a challenge on its own, often resulting in unsuccessful training of the agent. In this chapter, we therefore address the sparse reward problem in RL. Our goal is to find an effective alternative to reward shaping, without using costly human demonstrations, that would also be applicable to a wide range of domains. Hence, we propose to use model predictive control (MPC) as an experience source for training RL agents in sparse reward environments. Without the need for reward shaping, we successfully apply our approach in the field of mobile robot navigation both in simulation and real-world experiments with a Kuboki Turtlebot 2. We furthermore demonstrate great improvement over pure RL algorithms in terms of success rate as well as number of collisions and timeouts. Our experiments show that MPC as an experience source improves the agent's learning process for a given task in the case of sparse rewards.

3.1 Introduction

Reinforcement learning (RL) as well as model predictive control (MPC) have been applied lately to various fields and shown impressive results. However, there are still great challenges that need to be dealt with in both approaches. One major challenge in RL is the design of the reward function. Shaping the reward function to achieve desired results requires lots of trials to get the expected behaviour of the trained policy. This is mainly due to the fact that during the training, the agents exploit any opportunity given by the reward function. An obvious solution to this issue would be to use sparse rewards, i.e., rewarding the agent only for achieving the goal and giving zero rewards otherwise. While this approach encourages the agent to complete a certain task, it is more difficult for the agent to identify promising behaviour. Since the agent has no idea how well it is performing during the training before reaching the goal, it may fail to find the optimal policy. Handling sparse rewards has been an active topic in the field of reinforcement learning [14], [15], [29], [52], [53]. However, it still remains an open

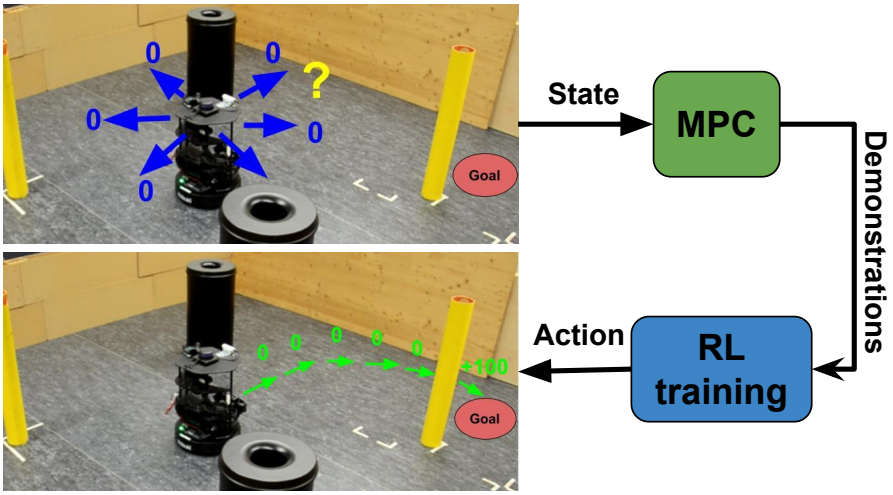


Figure 3.1: Handling sparse rewards using our approach. The model predictive controller (MPC) provides demonstrations to the reinforcement learning (RL) agent during training, while exploration still takes place. The MPC demonstrations in combination with RL guide the agent to find better policies to reach its goal.

question how an RL agent be successfully trained in a sparse reward setting using an approach that is applicable to a variety of domains.

One possibility is to use demonstrations to provide the agent with a course of actions that solve the task at hand. While demonstrations have been shown to improve the training process in case of sparse rewards [54], [55], [56], and human demonstrations specifically are commonly used in the literature, providing these demonstrations can be quite costly. In addition to that, human demonstrations typically require hardware equipment or virtual reality sets to provide the demonstrations [30], [31], [54]. In this work, we therefore propose to use MPC as an experience source for RL in the case of sparse rewards. MPC has been very popular lately in robotics and industry [32], [33], [57], [58], [59], [60] as it is able to handle constraints on both states and control signals, can handle multiple-input multiple-output systems as well as nonlinear systems, and the cost function can be constructed in a straightforward way by minimizing the deviation between the reference states and the current states. The aim of our work is therefore to show that MPC can be used to provide demonstrations for an RL agent in sparse reward settings.

The motivation of using RL with MPC demonstrations is as follows: First, MPC is computationally demanding since it solves an optimization control problem at each time step. For highly nonlinear models with numerous states, this may not be feasible to run in real time on real-world applications [32]. In contrast, inferring a trained policy online for actions is less demanding even for systems with large state spaces. Second, while MPC can be tuned to satisfying performance in a certain scenario, the performance will not be as satisfying when the same controller is deployed in another scenario. This becomes obvious in trajectory tracking, where the weight matrices have to be further tuned for different trajectories [61]. Third, unlike MPC which demands the full state of

the robot dynamic model, RL agents need only to attain partial observations from the environment which can be provided using onboard sensors [28].

We investigate our approach in the field of robot navigation for the following reasons: 1) For MPC, the kinematic prediction model of mobile robots is straightforward. 2) The state and action space of mobile robots is small in comparison to, e.g., humanoids, so that tuning of the MPC is not time-consuming. 3) The learned policy can easily be tested in different scenarios and usually successfully be transferred to a real mobile robot.

To summarize, our main contribution is to demonstrate that MPC as an experience source improves the training process of RL agents in sparse rewards settings. We showcase our approach in a mobile robot navigation scenario with static and dynamic obstacles, both in simulations and on a real robot. We also perform an ablation study to analyze the effect of varying the number of MPC demonstrations during the training. We make the following key claims:

(i) MPC guides the RL agent to learn tasks in a pure sparse reward setting.

(ii) The learned behavior policy leads to higher success rates than pure RL.

(iii) The balance between MPC demonstrations and RL exploration influences the convergence rate of the training.

(iv) Our approach can successfully be applied to the task of mobile robot navigation.

3.2 Related Work

3.2.1 Human Demonstrations in RL

Several approaches have been presented that use human expert demonstrations to boost the training process of RL agents by showing examples of how to perform a certain task. The agents subsequently learn faster in comparison to exploring randomly. For example, [62] used human demonstrations to boost the training of deep Q-networks and showed great improvement over different RL approaches in Atari games. A similar approach [29] used human demonstrations to improve the training of deep deterministic policy gradient (DDPG) in robotics tasks. Recently, [54] and [63] proposed combining supervised learning with RL and providing expert samples to play a video game and control a self-driving vehicle, respectively. Additionally, [30] also combined supervised learning with RL and human demonstrations to perform robot arm tasks and showed that using demonstrations outperforms the Hindsight Experience Replay (HER) approach [14].

3.2.2 Non-Human Demonstrations:

To overcome the need for costly human demonstrations, several approaches for providing non-human demonstrations have been proposed. [15] applied a hand-crafted policy of low success rate to improve the training of an unmanned aerial vehicle (UAV)

in a sparse reward setting. As stated in that work, these hand-crafted policies cannot be applied to diverse scenarios since they are only able to perform fixed maneuvers. [64] used a partially trained RL agent with shaped rewards to provide demonstrations for another RL agent in sparse reward settings on MuJoCo [65] simulations and to mobile robot navigation. [66], [67] proposed using proportional controllers to provide the demonstrations for RL agents for mobile robot navigation and robotic arm manipulation, respectively. [53] used demonstrations generated by a global planner to train a network using imitation learning along with RL for mobile robot navigation. Unlike the discussed approaches, we use a model predictive controller as an experience source since MPC can be applied to a variety of applications and does not involve reward shaping.

3.2.3 Combining MPC with RL

Several approaches using MPC and differential dynamic programming (DDP) along with RL have been presented. In [28] and [68] the authors implemented the guided policy search (GPS) approach where they transform the RL problem into a supervised learning problem using demonstrations from MPC and DDP respectively to train a UAV and MuJoCo environments. [69] used MPC as an experience source and trained their network using supervised learning for the navigation of a simulated car model. However, their approach keeps the MPC running as a safe fail policy in case the RL agent fails to find a better action than the MPC. In [70] the authors proposed to apply meta reinforcement learning along with MPC for demonstrations to train a mobile robot navigate through randomly moving obstacles. The authors used shaped rewards and the MPC is always running in case the agent cannot find an action specially when the robot is close to obstacles or the goal location. Furthermore, [71] trained the RL agent as a higher layer on top of the MPC to provide correction actions for the MPC to push objects using a robot arm. Unlike the previous approaches, we rely solely on RL training to learn from the MPC demonstrations and do not include supervised learning loss. Furthermore, we use a sparse reward setting to avoid reward shaping.

In [72] the authors applied DDPG after being trained offline to calculate a reference trajectory that is tracked by the MPC to control a Pendubot. [73] deployed imitation learning to learn from the MPC in cart-pole and autonomous driving scenarios. If the uncertainty of the network is high, the control is given solely to the MPC. To learn the threshold of uncertainty for switching the control the authors applied an RL agent. Furthermore, [34] trained an RL agent to generate subgoals that are tracked using a MPC controller. This approach was implemented in a robot navigation scenario. Despite the improved performance of these approaches over pure RL methods, all of them require the MPC to be running the whole time which can be computationally demanding. In our work, we only use the MPC during the training process. During testing solely the learned RL policy is applied without the need for MPC as a fallback policy.

To the best of our knowledge, using MPC as an experience source to improve the

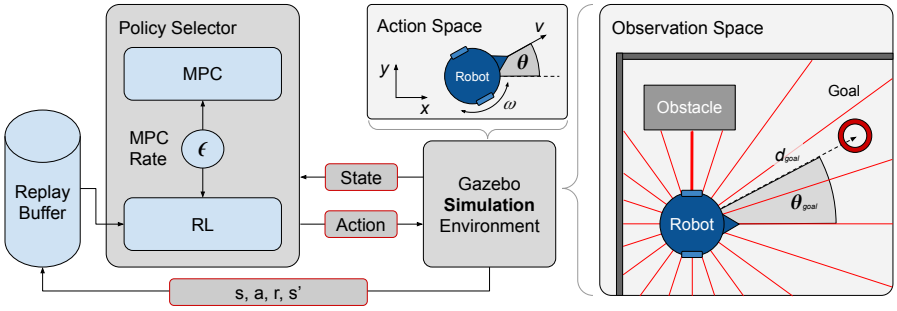


Figure 3.2: Illustration of our approach. **Left:** Both the model predictive control (MPC) and reinforcement learning (RL) interact with the simulated environment during training. Based on the MPC rate ϵ, the policy selector chooses between MPC demonstrations and RL. The gathered tuples (s, a, r, s') are stored in the replay buffer. **Top middle:** The robot's kinematics are used for nonlinear MPC. **Right:** The observation space includes a laser scan (red lines), the closest obstacle (thick red line), and the relative distance and heading to the goal.

training of RL agents in the case of sparse rewards has not been tackled before.

3.3 Our Approach

The goal of our work is to run the RL agent independently of the MPC after training, while relying on the MPC demonstrations before. During training, we use a parallel architecture where the MPC and the RL agent run simultaneously and only one of the two output actions is chosen. To demonstrate our approach, we focus on solving mobile robot navigation around obstacles using only the laser scan as input to the RL agent. We run the chosen policy for the whole episode to provide demonstrations showing how the robot can navigate from the start position to the goal while avoiding obstacles. To choose between the MPC action and the RL action, we define the MPC rate ϵ that determines whether the MPC output or the RL action is taken for the whole upcoming episode. In the following, we present the components of our architecture in detail, a schematic overview can be found in Figure 3.2.

3.3.1 Nonlinear Model Predictive Control

The nonlinear model predictive control (NMPC) is a variant of the MPC that can handle nonlinear systems. In MPC schemes, there are two main components: the prediction model and the optimal control problem (OCP). The prediction model is the robot dynamics model that is used to calculate the future states of the robot in a receding horizon fashion, i.e., the next future states of the robot are estimated at each new time step. The OCP consists of the cost function that has to be minimized while respecting the control and state constraints of the robot. To achieve real-time solvability, we implement the controller using acados [74], which leverages the multiple shooting transcription method (see Sec. 2.3.4.2) to efficiently solve the resulting nonlinear optimization problem.

Parameter	Value
Q	diag([0.1, 0.1, 0.05])
Q_e	diag([1, 1, 0.1])
R	diag([0.001, 0.01])
D	20
N	30
T	6 seconds

Table 3.1: MPC parameters and weight matrices. Q_e is the weight matrix at the final state N. T is the prediction horizon in seconds.

3.3.1.1 Prediction Model

The choice of the prediction model is critical for the design of the NMPC. A high fidelity model can accurately represent the nonlinearities of the actual robot but at the cost of high computational cost. While a simple mathematical model can lower the computational cost but at the cost of less accurate predictions. In this work, we represent the mobile robot using a nonlinear kinematic model (Figure 3.2). The state space representation of the model $\dot{X} = f(X, A)$ is as follows:

$$\begin{aligned} \dot{x} &= vcos(\theta) \\ \dot{y} &= vsin(\theta) \\ \dot{\theta} &= \omega \end{aligned} \tag{3.1}$$

where X represents the state of the mobile robot, i.e., its 2D position (x, y) in space, and the heading of the robot θ. A represents the controls which are the linear and angular velocities, v and ω, respectively.

3.3.1.2 Optimal Control Problem (OCP)

The OCP solved at each time step is formulated by setting the cost function and the constraints on the controls and the states and is as follows:

$$\min_{\substack{x_{k=1:N},\\ a_{k=1:N-1}}} \quad J = \sum_{k=1}^{N} \|x_{ref} - x_k\|_Q^2 + \|a_k\|_R^2 + \|\frac{1}{e^{dist_k^2}}\|_D^2 \tag{3.2a}$$

$$\text{subject to} \quad x_{0|k} = x_k, \tag{3.2b}$$

$$x_{i+1|k} = f(x_{i|k}, u_{i|k}), \tag{3.2c}$$

$$a_{i|k} \in A, \tag{3.2d}$$

$$x_{i|k} \in X \tag{3.2e}$$

Where J represents the cost function to be minimized. The first term penalizes the difference between the predicted states x_k and the target state x_{ref} at instant k. The second term penalizes the control signals a_k. The final term penalizes the nearness to the closest obstacle, where $dist_k$ is the Euclidean distance to the nearest obstacle at in-

Parameter	Value
optimizer	Adam
discount factor (γ)	0.99
replay buffer size	10^6
hidden layers (all networks)	2
hidden units per layer	256
batch size	256
learning rate	3.10^{-4}

Table 3.2: Parameters for SAC

stant k. Q and R are diagonal weight matrices for the states and controls respectively. The weighting factor D balances the collision avoidance term. Constraint (3.2b) defines the initial state, (3.2c) forces the system's dynamic constraints, while (3.2d) and (3.2e) ensure that the controller satisfies the control and state limits at each time step. The weight matrices along with the MPC parameters are shown in Table 3.1. Note that the cost function uses the full state of the robot, while the RL uses only partial observations. Hence, the cost function is not usable as a reward function and the reward for the RL agent is still sparse.

3.3.2 Reinforcement Learning Agent

The foundation of reinforcement learning relies on the description of the world as a Markov decision process (MDP), which is described by a tuple M: (S, A, R, P, γ). Where S is the set of states, A is the set of actions, $R(s, a)$ is the reward function, $P(s'|s, a)$ is the trasition probability, and γ is the discount factor. An agent in state $s \in S$ takes an action $a \in A$ resulting in the next state $s' \in S$, which is rewarded by reward r and dsicounted by factor γ. The action a is chosen according a policy π that determines for each state which action the agent will take. The transition from state s to state s' upon taking action a is determined by the transition probability P which is also environment dependent. The main goal of the RL agent is then to maximize the total cumulative reward:

$$R_{\text{total}} = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{3.3}$$

We use an off-policy algorithm to control the mobile robot and use soft actor-critic (SAC) [75] as the main agent. SAC maximizes the entropy of the policy along with the reward which resulted in a better exploration strategy and showed improved convergence rates compared to other algorithms in our experiments.

3.3.2.1 Soft Actor-Critic

The soft actor-critic aims to maximize the expected reward and the entropy of the policy by optimizing the following objective function:

$$J = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t) + \alpha H(\pi(.|s_t))], \tag{3.4}$$

where r is the reward the agent gets for executing action a_t at state s_t, π is the policy, ρ_π is the trajectory distribution induced by the policy π, α is the temperature of the entropy H, and $H(\pi(.|s_t)) = -E_\pi[\log \pi(.|s_t)]$ is the entropy of the policy.

We used the same architecture as in [75] and also implemented the entropy regularization by optimizing α during the training. Our hyperparameters for the SAC are summarized in Table 3.2.

3.3.2.2 MPC Rate

The MPC rate ϵ refers to the probability of choosing the MPC actions over the RL actions. Hence, this parameter can greatly affect the training performance since a high contribution of the MPC would mean less exploration of the RL agent and would result in a policy that acts more similarly to the MPC but is limited by the scenarios that only the MPC controller has experienced. A small ϵ would lead to more exploration and less demonstrations from the MPC. In our experiments, we provide an ablation study regarding different values for this parameter and its effect on the training. We found it beneficial to decay the influence of the MPC over the course of training. Hence, we decay ϵ over episodes by a decay rate of 0.5% as the episodes (n) progress:

$$\epsilon_n = \epsilon_0 (0.995)^{n/4} \tag{3.5}$$

Where ϵ_0 is the initial value for the MPC rate at the beginning of the training and ϵ_n is the updated MPC rate.

3.3.3 Implementation of the RL Agent

For the implementation of the RL agent in the navigation task, we defined the following action space, observation space, and reward function (see also Figure 3.2).

Action space: We use continuous action spaces for both the linear and angular velocities of the mobile robot. The limits for the linear and angular velocities are $[-0.5\,m/s, 0.5\,m/s]$ and $[-0.785\,rad/s, 0.785\,rad/s]$, respectively.

Observation space: At each time step, the RL agent receices a scan of 20 lidar distance measurements, as well as the distance and heading difference to the target location as input. We furthermore provide the explicit distance and heading to the nearest obstacle as calculated from the laser scan.

Rewards: The sparse reward is defined as follows:

$$R = \begin{cases} r_{success} & \text{if goal reached,} \\ r_{collision} & \text{if collision or stuck,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

3.4 Experimental Evaluation

The main focus of this work is to show how MPC can improve the training of RL agents in a sparse reward setting. We use Gazebo [76] in combination with ROS [77] as a simulator during training and evaluation, and as a real-robot platform the Kuboki Turtlebot 2. We applied an MPC rate of 25% during the experiments. To show that the MPC guides the RL agent to reach better behavior policies than pure RL in a sparse reward settings, and that our approach can be applied to mobile robot navigation, we evaluate our approach in the following scenarios: (i) A static environment, where the robot has to navigate around obstacles to reach different target positions. (ii) A dynamic environment, where four dynamic obstacles are rotating in the environment and the robot has to navigate between them to reach the target. (iii) We test whether our agent is able to generalize to unseen scenarios, i.e., we trained the agent in different environments and evaluate its performance in a further, unseen environment. (iv) Finally, we evaluate the learned policy on a real robot. The experiments can be seen in the video[1].

3.5 Experimental Modeling Assumptions

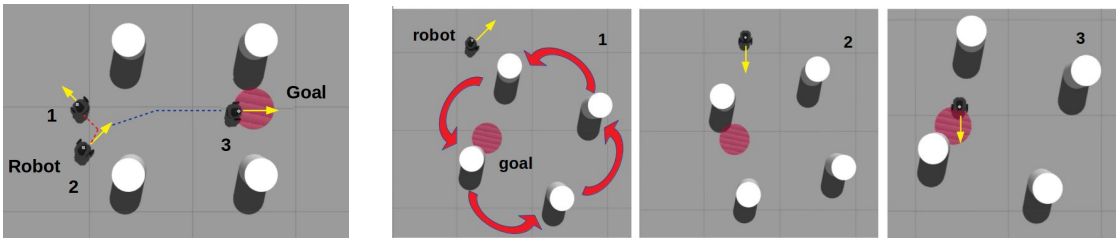
The following modeling assumptions establish the specific configuration of the system dynamics, observation spaces, and safety constraints employed in this chapter's scenarios:

- **System Dynamics**: The robot is modeled as a discrete-time kinematic unicycle. The control loop for both the reinforcement learning agent and the MPC teacher operates at 20 Hz.
- **Observation Space**: The state representation includes 20 equiangular LiDAR rays providing a 360° field of view. These sensor readings are raw distance measurements used without normalization or clipping. To obtain these 20 readings, the LiDAR scan is divided into 20 equidistant sections, and the minimum distance value from each section is extracted to represent the local obstacle proximity.
- **Safety Constraint Definitions**: In this chapter, safety is defined by the hard kinematic and collision-avoidance constraints enforced within the MPC optimization to ensure the feasibility of training demonstrations. Additionally, the RL agent is subject to a sparse penalty in the reward function in the event of a collision to discourage unsafe behaviors discovered during exploration.

3.5.1 Static Environment

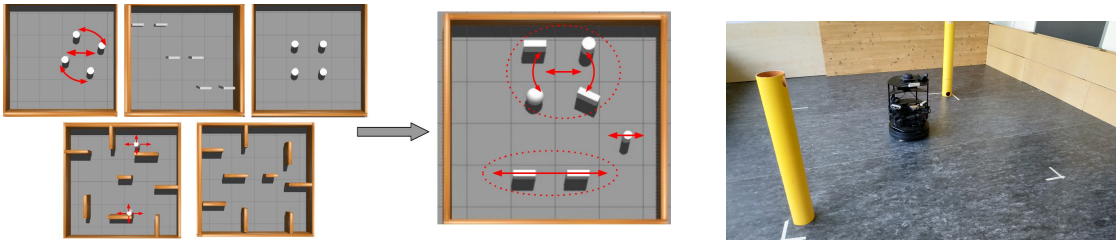
The first experiment shows the performance of our approach in the case of static obstacles. The environment is shown in Figure 3.3a. We tested our approach with soft actor

[1] `https://youtu.be/Au6R92JHH5Q`



(a) Static environment (b) Dynamic environment

Figure 3.3: Birds-eye view of the MPC_SAC agent navigating from a random position to the goal (round red patch) in case of **(a)** static obstacles and **(b)** dynamic obstacles, both represented by white cylinders. The yellow arrows indicate the heading of the robot at different instances, the red trail indicates backward motion, while the blue trail indicates forward motion. The numbers denote the temporal order.



(a) Generalization test (b) Real-world environment.

Figure 3.4: **(a)** Generalization test: The robot is trained in the five left environments and is evaluated in the right one. The evaluation environment has moving walls and spheres which the agent has not jointly experienced during training. **(b)** The real environment for testing the Turtlebot has a size of $3.7 \times 3.7\,m^2$ with two static obstacles. The goals are spawned randomly around the area.

critic (SAC) as described before and, additionally, with twin delayed deep deterministic policy gradient (TD3) [78] to show the general applicability. We trained the agents with and without the MPC for 7,000 episodes, where each episode ran for 1,000 steps. During each episode, we spawned a new random goal every time the previous goal had been reached. An episode terminates if the agent collides with an obstacle or if the robot is stuck in place for 20 consecutive steps.

As a further baseline, we implemented guided policy search (GPS) [28], [68] and trained a deep neural network in a supervised learning manner. We use the partially observed state (same state used by the RL agents) as the input to the network to predict the actions and use the actions from the MPC to calculate the mean squared error loss for the network. Simultaneously, we minimize the deviation of the MPC action from the inferred action at each time step.

In Figure 3.5a we show the convergence curves for our approach with SAC (cf. MPC_SAC) plotted against the pure SAC. As can be seen, our MPC_SAC shows better performance than the pure SAC in terms of collected reward.

For evaluation of the trained agents, we spawned a fixed sequence of 70 random goals and calculated the success rate, collision rate, and timeout rate, the results are shown in Table 3.3 and demonstrate that the MPC successfuly guides the RL agent to learn tasks

Static Environment			
Agent	Success rate	Collisions	Timeout
SAC	52.5%	14.7%	32.8%
MPC_SAC	**97.2%**	0%	2.8%
TD3	34.4%	2.8%	62.8%
MPC_TD3	**98.6%**	1.4%	0%
GPS	88.5%	0%	11.5%
Dynamic Environment			
SAC	42.8%	34.4%	22.8%
MPC_SAC	**91.4%**	7.2%	1.4%
GPS	82.8%	4.3%	12.9%
Generalization to Different Environments			
MPC_SAC	68.6%	26.7%	4.7%
Real-Robot Experiments			
MPC_SAC	$85 \pm 5\%$	$6.7 \pm 2.3\%$	$8.3 \pm 2.3\%$

Table 3.3: Evaluation in different test scenarios. For each scenario, a fixed sequence of random goals was spawned. Timeout is the fraction of goals that the agent missed to reach within a fixed number of time steps. Our approach using the MPC experience outperforms both pure RL (SAC[75] and TD3[78]) and GPS (Guided Policy Search)[68] approaches.

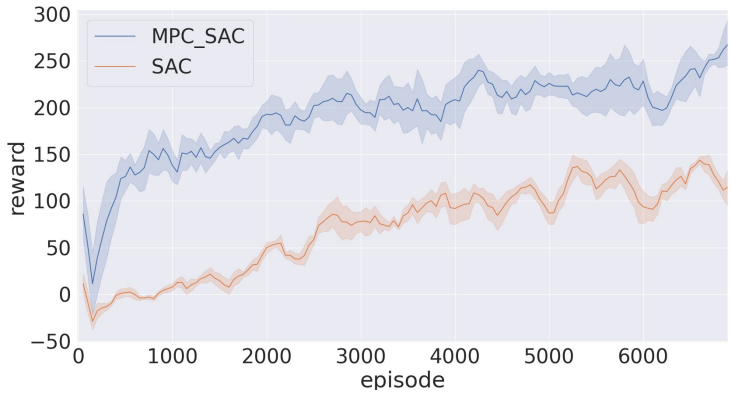
in a sparse reward setting and that the learned policy has higher success rates than pure RL. The SAC on its own was able to reach the targets if it is spawned directly in front of the robot, but fails to navigate through the obstacles to reach further targets. Introducing the MPC to the training process makes the agent able to navigate around the obstacles to reach its targets. While the pure TD3 was stuck most of the time, with the MPC it was able to reach an impressive performance, i.e., the agent showed higher success rates as well as fewer collisions and timeouts. Also, GPS outperforms the pure RL agents, however, our approach shows superior performance compared to GPS, which can be credited to the exploration involved in the RL training.

3.5.2 Dynamic Environment

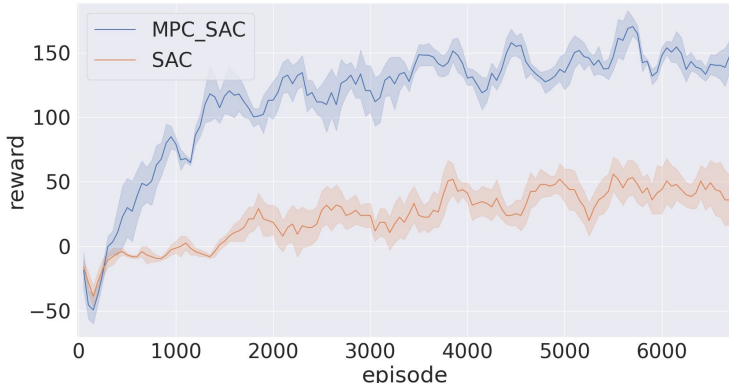
In the second experiment, we tested our approach in scenarios with dynamic obstacles, i.e., four obstacles rotating in the environment (see Figure 3.3b) where the robot has no information about the behavior of the obstacles. The training performance shown in Figure 3.5b indicates that including the MPC improves the navigation performance per episode by one goal (+100 reward) on average.
This improvement is also reflected in the success rate during testing, which increased by almost 50% (see Table 3.3), indicating the improvement achieved by introducing the MPC as demonstrations during training.

3.5.3 Generalization to Different Environments

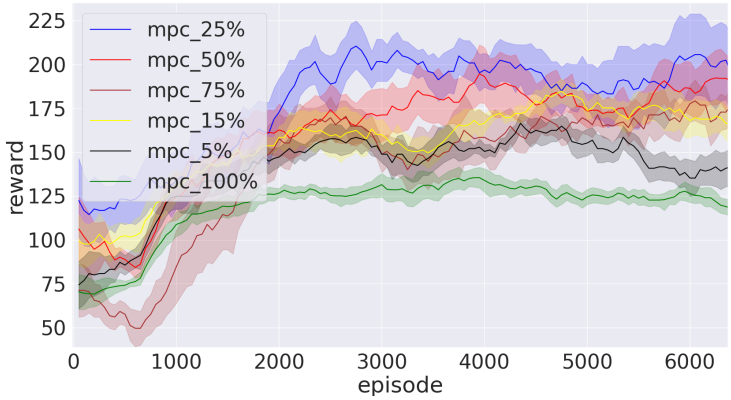
The aim of this experiment is to test whether the demonstrations provided by the MPC can aid the RL agent to handle unforeseen scenarios. We trained the agents in five different environments to gather diverse experience and then evaluate its performance in a sixth environment (see Figure 3.4a). The results are also included in Table 3.3. The agent was unable to handle obstacles not experienced before such as moving walls and spheres, however, we found that the robot learned a primitive strategy that enables it to reach its target at some instances. More specifically, the agent waits for the moving



(a) SAC vs MPC_SAC in static environment (Figure 3.3a).



(b) SAC vs MPC_SAC in dynamic environment (Figure 3.3b).



(c) Ablation study for ϵ_0 in static environment

Figure 3.5: Training performance for different setups. **(a)**, **(b)** Including MPC as an experience source significantly increases the training performance by means of collected reward. **(c)** We found an initial MPC rate of 25% to give the best results. The bold lines show the mean of five different seeds while the shaded areas as the standard error.

obstacles until they do not block the way towards the goal anymore.

3.5.4 Real-Robot Experiment

Finally, we demonstrate that our approach is transferable to real mobile robots. We trained an agent in the static environment. To improve the sim-to-real transfer in terms

of sensor noise, we added Gaussian noise ($\mu = 0, \sigma = 0.15$) to the lidar observation and Gaussian noise to the relative distance to the goal during training. The environment for the real-robot experiments is shown in Figure 3.4b and the mean success rate for three trials with 20 random goals each is included in Table 3.3. The robot had a mean success rate of 85%, a mean collision rate of 6.7% and a mean timeout rate of 8.3%, over 3 runs.

3.5.5 Ablation Study

The balance between RL and MPC influences the training performance. Hence, we conducted an ablation study to evaluate different MPC rates ϵ_0 for the contribution of the MPC controller. We investigate how including more or less MPC episodes affects the training of the RL agent (see Figure 3.5c), i.e., we evaluated the performance for $\epsilon_0 = [5\%, 15\%, 25\%, 50\%, 75\%, 100\%]$. Note that those are the initial rates, which decay during training, see Equation (3.5). We experimentally found that a MPC rate of 25% gave the best results in terms of reached rewards. This is a similar to [70], where an activation rate for the MPC of 20% was found to be best.

3.6 Conclusion

In this chapter, we presented a novel approach to handle the challenges of sparse rewards in reinforcement learning (RL) using model predictive control (MPC) as an experience source. We show that the MPC demonstrations guide the RL agent to converge faster and find better policies. In the domain of robot navigation, our approach outperforms pure reinforcement learning algorithms in terms of success rate as well as number of collisions and timeouts. Furthermore, our ablation study shows the effect of varying the MPC rate on the training result. Finally, we showed that the learned controller can be successfully applied to a real robot. The results presented in this chapter demonstrate that leveraging MPC as a synthetic experience source successfully resolves the sparse reward bottleneck for single-robot navigation, fulfilling the requirements for the efficient, policy-only deployment outlined in RQ1.

4 Safe Multi-Agent Reinforcement Learning for Behavior-Based Cooperative Navigation

Abstract

Having addressed single-agent efficiency, we now extend this framework to multi-robot systems, where the role of MPC shifts from an experience source to an online safety filter to enable scalable formation control. In this chapter, we address the problem of behavior-based cooperative navigation of mobile robots using safe multi-agent reinforcement learning (MARL). Our work is the first to focus on cooperative navigation without individual reference targets for the robots, using a single target for the formation's centroid. This eliminates the complexities involved in having several path planners to control a team of robots. To ensure safety, our MARL framework uses model predictive control (MPC) to prevent actions that could lead to collisions during training and execution. We demonstrate the effectiveness of our method in simulation and on real robots, achieving safe behavior-based cooperative navigation without using individual reference targets, with zero collisions, and faster target reaching compared to baselines. Finally, we study the impact of MPC safety filters on the learning process, revealing that we achieve faster convergence during training and we show that our approach can be safely deployed on real robots, even during early stages of the training.

4.1 Introduction

Cooperative navigation of unmanned vehicles has gained considerable attention due to its applications in missions such as search and rescue [79], surveillance [80], and payload transfers [81]. As outlined in the literature [82], cooperative navigation can be achieved through various strategies, including leader-follower, virtual-structures, and behavior-based approaches. Behavior-based methods [83], [84] offer more flexibility, as the robots' actions adapt to their own observations. The goal in behavior-based cooperative navigation is to maintain relative distances, avoid collisions, and reach target locations.

In this work, we focus on safe learning for behavior-based cooperative navigation. Safety in this context means eliminating all collisions during both training and execution, achieved through safe reinforcement learning (RL). RL has been successful in cooperative navigation [85], [86], [87], but challenges remain due to the sim-to-real gap. This gap arises from real-world sensor noise and unmodeled nonlinearities in simulations. Unlike approaches that focus on improving simulation realism [88], [89], we emphasize the need to ensure RL is safe for real-world deployment. Although safety in RL has advanced [90], [91], [92], it remains underexplored in behavior-based navigation. Furthermore, the impact of safety filters on RL training has not been fully studied.

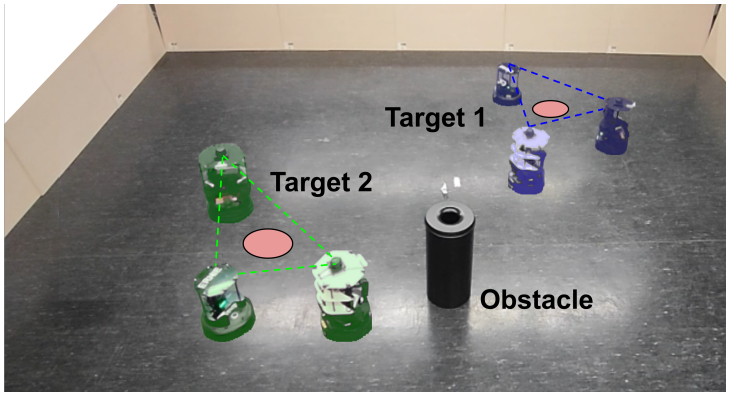


Figure 4.1: Real-world example for the behavior-based cooperative navigation control. The robots start from random locations and navigate cooperatively to reach the targets for the centroid of the formation (shown in red) while aiming to maintain the predefined distances with respect to each other. The blue and green shades show the robots team at the first and second goals, respectively.

In this work, we investigate the effects of applying a model predictive control (MPC)-based safety filter to RL in the context of behavior-based cooperative navigation.

To facilitate the deployment of behavior-based cooperative navigation and enable the scalability of our approach, i.e., the application of the learned policy to a team with a higher number of robots without retraining, we use a reference target for the centroid of the formation. This modification is promising for coordinating teams of robots as in the StarCraft multi-agent challenge [93]. To maintain the distances between the robots, each robot considers the distances to its two neighbors. Optimization-based controllers struggle in solving this task, since these controllers require a reference target location for each individual robot and might even assume that each robot has access to all its neighbors positions [83], [84], [94], [95], [96], [97], [98], [99]. Therefore, we use multi-agent reinforcement learning (MARL) to achieve the desired behavior. Figure 4.1 shows a real-world example of our approach, where the robots get into formation to reach the target locations for the centroid while avoiding unsafe actions.

The primary contributions of this chapter can be summarized as follows:

(i) **Safe learning of behavior-based cooperative navigation.** We introduce the application of safe RL to behavior-based navigation, ensuring agents' safety during training and execution—a previously unaddressed challenge. We also study the effect of the safety layer on training efficiency, showing improved convergence compared to baselines.

(ii) **Eliminating the need for individual path planners.** We study robot behavior in the cooperative navigation task to use less information, yielding a policy adaptable to more robots without retraining. Our approach requires only a single reference target for the centroid, a setup not previously addressed in MARL.

(iii) **Behavior-based navigation on real robots.** We demonstrate the learned policy on real robots. Distributed MPC safety filters ensure zero collisions throughout

the experiments, and training on real robots remains safe, eliminating collisions even in early stages. The experiments can be seen in the video[2]

4.2 Related Work

4.2.1 Cooperative Navigation Using RL

Several studies applied RL to achieve cooperative navigation using different formulations. In [85] the authors used multi-agents proximal policy optimization[100] along with curriculum learning, reference targets, and relative positions of the robots with respect to each other. Additionally, [97] applied PPO for formation control of quadrotors, incorporating individual goals for each unit. The authors in [86] used imitation learning along with RL to formulate a distributed formation controller based on a leader-follower scheme. In [87] the authors used policy gradients along with a graph convolutional network (GCN) and reference targets for each robot to achieve static formations of drones. [101] and [102] achieved formation control of maritime unmanned surface vehicles using deep deterministic policy gradient (DDPG) and deep Q-networks, respectively, based on leader-follower approaches. Moreover, [103] proposed a motion planner based on a multi-agent extension of the soft-actor-critic (SAC) to coordinate multiple robots to reach their respective goals. **Different from the previous works**, we only use reference targets for the centroid of the formation and information about only two neighbors for each robot. More importantly, we ensure collision-free training by using MPC-based safety filters, and we use the attention mechanism to account for the interaction between the robots.

4.2.2 Safety in MARL

Safety is a crucial aspect in MARL, with ongoing developments addressing this concern. [104] utilized control barrier functions (CBF) [105] along with multi-agent DDPG (MADDPG) for collision-free navigation of two agents. [90] explored the use of attention modules and decentralized safety shields based on CBF to reduce collisions in autonomous vehicle scenarios. Similarly, [91] implemented a centralized neural network as a safety layer with MADDPG in cooperative navigation. Moreover, [106] proposed using a predictive shielding approach, along with MADDPG, to ensure the safety of multi-agents in cooperative navigation scenarios. [107] presented RL-based model predictive control to achieve leader-follower formation control of mobile robots while ensuring their safety. **In contrast to the previous works**, we focus on behavior-based navigation where collisions between agents during navigation are more likely to occur, since the agents are required to navigate close to each other and not just have a single instance where their paths intersect. Additionally, the previous studies focused on the task of having individual reference targets, which can be achieved by training a single

[2] `https://youtu.be/ViVblAuagVE?si=eK9nKccwq4JgVV1h`

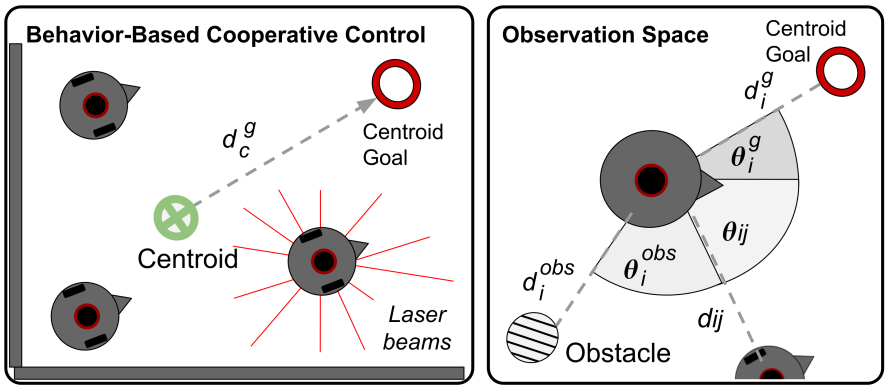


Figure 4.2: The observation space per robot includes lidar readings (red lines), distances and headings to the goal (d_i^g, θ_i^g), two neighbors (d_{ij}, θ_{ij}), and the closest obstacle $(d_i^{obs}, \theta_i^{obs})$. Additionally, robots have information about the centroid's distance to the goal (d_c^g) d_c^g.

RL policy and then deploy it to several robots as in [34], [35], [36]. Our task, on the other hand, necessitates using a MARL framework so that the robots interact with each other during the training to rely only on a reference location for the centroid. While the previous studies fell short of real-world experiments, we conducted extensive experiments with both trained and untrained policies on real robots to demonstrate the safety of our framework, and the performance of our behavior-based modification.

4.3 Problem Statement

We consider the task of safely training a team of mobile robots to move the centroid of their formation to a desired target location while avoiding collisions with static obstacles and neighboring robots, and aiming to maintain predefined distances between robots. We assume that each robot is equipped with a lidar sensor to perceive its surroundings, each robot can localize itself within the environment, and that there is no pre-knowledge about the environment or existing obstacles. To ensure the safety of the robots at all times, each robot has its own safety layer to avoid collisions with its neighbors and the obstacles. The robots do not communicate with each other, but navigate cooperatively based on information about their centroid and the reference target, and the relative distances. Each robot receives relative distances and angles only about its two neighbors even if the team consists of more robots, as well as the target location of the centroid, as illustrated in Fig. 4.2. Finally, each robot calculates the distance and relative angle to the closest obstacle from the lidar scan. We do not assume that the global information is available for each robot. This enables the scalability of our approach when applied to more robots, without retraining the policy.

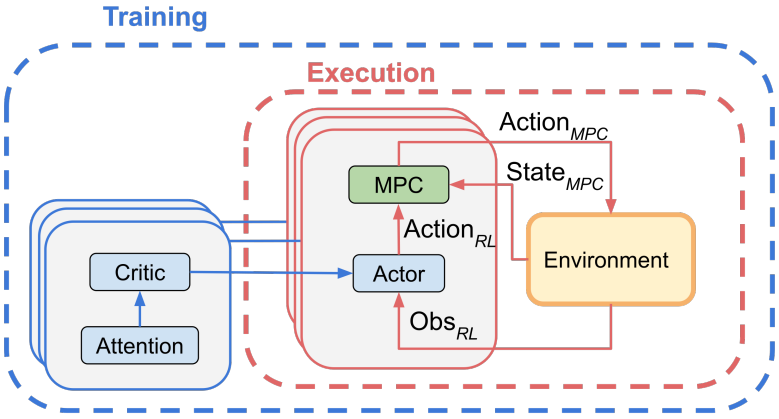


Figure 4.3: Centralized training decentralized execution (CTDE). At each time step, the agent interacts with the environment to receive the current observation Obs_{RL}, including relative information about the two neighbors and the closest obstacle, and outputs the Action$_{RL}$. The MPC controller receives the State$_{MPC}$, overrides any unsafe actions to prevent collisions, and sends Action$_{MPC}$ to the robot. During training (blue dashed), critics share all agents' observations, while during execution (red dashed), each actor accesses only its own observation.

4.4 Our Approach

In this section, we formulate our work as a MARL problem, introduce the attention module for agent interaction, and describe the MPC-based distributed safety filters in detail.

4.4.1 Multi-Agent Reinforcement Learning (MARL)

In this work, we adopt the Markov games' framework [108], which is formulated as a set of partial observable Markov decision processes (POMDPs). For the case of N agents, we have a set S that represents the augmented state space of all the agents, a set of actions $A_1, ..., A_N$ and a set of observations $O_1, ..., O_N$ for each agent. At each time step t, an agent i, receives an observation correlated with the state $o_i^t : S \rightarrow O_i$, which contains partial information from the global state $s^t \in S$. Where s^t refers to $state_{RL}$ in Fig. 4.3 . The agent chooses an action a_i^t according to a policy, $\pi_i : O_i \rightarrow A_i$, which maps each agent's observation to an action. The agent then receives a reward as a function of the state and the action taken, $r_i : S \times A_i \rightarrow \mathbb{R}$. For each agent, the tuple containing the state, the action, the reward, and the next state is added to the respective replay buffer $\mathcal{D}_i$. Our approach is based on the soft-actor-critic (SAC) [75] algorithm. We extend SAC to the multi-agent domain and apply the centralized-training decentralized execution (CTDE) [109] scheme. During *training* all the agents share their observations with each other. However, during *execution*, each agent has access to its own observation only as shown in Fig. 4.3.

For each agent we define the **observation space** as in Fig. 4.2. The observation includes 40 equiangular lidar readings, the relative distances d_i^j and angles θ_i^j to its two neighbors, the relative distance d_i^{goal} and angle θ_i^{goal} to the goal of the centroid, the dis-

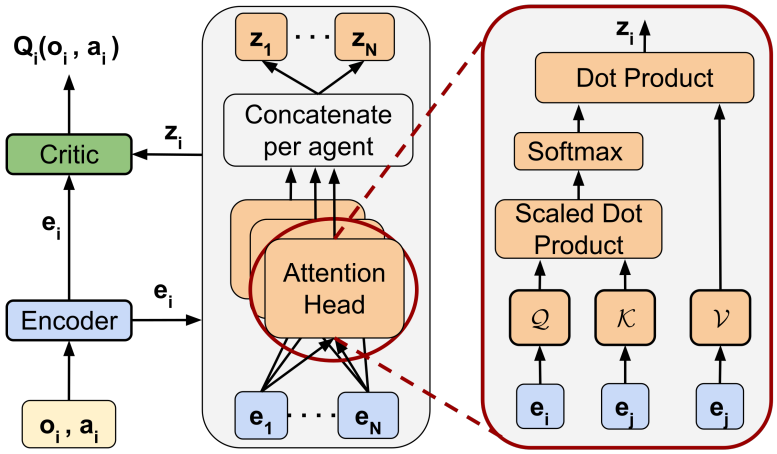


Figure 4.4: Attention-based multi-agent reinforcement learning. In the attention-based critics, observations are encoded separately and fed into attention heads to calculate weights based on the query and the key-value pairs. The attention output is then concatenated with the states and fed into the critics.

tance from the centroid of the formation to the goal $d^{goal}_{centroid}$, the relative distance d^{obs}_i and angle θ^{obs}_i to the nearest obstacle (calculated from the lidar scan), the desired reference distance between the robots, and the actions of the robot for the previous time step a^{t-1}_i which is the pair (v^{t-1}_i, w^{t-1}_i). For the **action space**, we control the linear and angular velocities v and w.

The **reward** function for each robot i is defined as:

$$\begin{aligned}\mathbf{r}^{\mathbf{t}}_{\mathbf{i}} = \mathbf{r}_{goal} \cdot \mathbf{1}_{\text{goal reached}} + \mathbf{r}^{collision}_i \cdot \mathbf{1}_{\text{collision or stuck}} \\ + (\mathbf{r}^{formation}_i + \mathbf{r}^{obs}_i + \mathbf{r}^{goal}_{centroid}) \cdot \mathbf{1}_{\text{otherwise}}\end{aligned}$$

where r_{goal} and $r^{collision}_i$ are sparse rewards indicating whether the centroid reaches the target and whether the robot collides or remains stuck for several steps, respectively. $r^{formation}_i$ is a reward for maintaining formation, computed as the negative error between the reference distances and actual distances among agents. r^{obs}_i is a reward for maintaining a safe distance from the closest obstacle, given by the negative obstacle distance. $r^{goal}_{centroid}$ is a reward for guiding agents toward the goal, calculated as the negative distance between the centroid and the goal.

4.4.2 Attention-Based Critics

To effectively capture relative information among agents, we implement an attention module (Figure 4.4) following the framework of [46], which computes attention based on queries, keys, and values. Each agent's observation is encoded into embeddings $e_1 \dots e_N$ by an encoder network. These embeddings are input to the K and V networks to generate keys and values, respectively, while the querying agent's embedding is processed by the Q network. The computed attention weights are then used to enhance the encoded observations, which are concatenated with the outputs z_i from the attention

module and input into the critics to compute Q-values.

4.4.3 Model Predictive Safety Filter

To ensure the safety of the agents, we implement distributed shields based on nonlinear model predictive control (NMPC). The MPC utilizes a mathematical model of the robot (4.1) to calculate the predicted states based on the current state and action. At each time step, the MPC solves an optimization problem (5.2) to find the optimal control sequence that satisfies a set of predefined constraints on the states and actions. Only the first action of the control sequence is applied while the rest of the sequence is discarded. Prior safety related studies have employed the MPC as a safety filter [106], [110], [111] due to its capability to handle systems with multiple states, controls, and constraints. To minimize the intervention of the MPC-safety filter and avoid disrupting the exploration of the RL agent, the mentioned studies align the MPC actions with those of the RL agent. In our work, we additionally, penalize the RL agent for deviating from the MPC safe actions, which in turn teaches the RL agent to not rely on the MPC-safety filter as we show in Sec.4.6.3.

4.4.3.1 Safety Filter Formulation

The robot's state in NMPC ($State_{MPC}$ in Fig. 4.3) is defined as $\mathbf{x} = [x, y, \theta]^T$, representing its position and heading. Controls are denoted by $\mathbf{a} = [v, \omega]^T$, matching the RL actions. A weighted Euclidean norm with a positive definite weighting matrix R is denoted as $||\mathbf{x}||_R^2 = \mathbf{x}^T R \mathbf{x}$.

4.4.3.2 Prediction Model

The discrete-time model of the robot is:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t) = \mathbf{x}_t + \begin{bmatrix} cos\theta_t & 0 \\ sin\theta_t & 0 \\ 0 & 1 \end{bmatrix} \mathbf{a}_t \Delta t \tag{4.1}$$

4.4.4 Optimal Control Problem

To ensure the safety of the robot while minimizing the intervention of the safety filter, we formulate the optimization problem for each robot i at time step t as follows:

$$\min_{\substack{\mathbf{x}_{t:t+T|t},\\ \mathbf{a}_{t:t+T-1|t}}} \|\mathbf{a}_{RL} - \mathbf{a}_{t|t}\|^2_{R_0} + \sum_{k=1}^{T-1} \|\mathbf{a}_{t+k|t}\|^2_R + \sum_{k=0}^{T} \| \frac{1}{e^{dist_{i,1}^{t+k|t}}} \|^2_D$$
$$+ \sum_{k=0}^{T} \| \frac{1}{e^{dist_{i,2}^{t+k|t}}} \|^2_D + \sum_{k=0}^{T} \| \frac{1}{e^{dist_{obst}^{t+k|t}}} \|^2_D \tag{4.2a}$$

$$\text{s.t.} \quad \mathbf{x}_{t|t} = \mathbf{x}_t, \tag{4.2b}$$

$$\mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{a}_{t+k|t}), \forall k = 0, 1, ..., T-1 \tag{4.2c}$$

$$\mathbf{a}_{t+k|t} \in \mathbf{A}, \qquad \forall k = 0, 1, ..., T-1 \tag{4.2d}$$

$$\mathbf{x}_{t+k|t} \in \mathbf{X}, \qquad \forall k = 0, 1, ..., T-1, \tag{4.2e}$$

where T represents the prediction horizon. The notation $t+k|t$ indicates predictions at time $t+k$, assuming the current time is t. The optimization terms in (4.2a) are structured in three main components. The first term measures the deviation between the RL agent's proposed action $\mathbf{a}_{RL}$ (Action$_{RL}$ in Fig. 4.3) and the initial MPC action $\mathbf{a}_0$ (Action$_{MPC}$), optimizing for minimal discrepancy. The second term minimizes the magnitude of future control signals $\mathbf{a}_k$, promoting smoother transitions. Third, the distances $dist_{i,1}$ and $dist_{i,2}$ between the robot and its two neighbors, as well as $dist_{obst}$ to the nearest obstacle, are penalized to ensure safety. The exponential function applied to distances has proven effective in enhancing obstacle avoidance, as shown in previous research [112]. The weight matrices R_0, R, and D are tuned such that R_0 has higher weights than R to ensure that the first action matches the proposed action $\mathbf{a}_{RL}$. D has the highest weight to maximize the cost on getting close to obstacles.

Constraints (4.2b–4.2e) ensure adherence to system dynamics and operational limits, defined as follows: (4.2b) sets the initial state, (4.2c) enforces the robot's dynamic model from Eq. (4.1), (4.2d) and (4.2e) maintain the actions and states within predefined bounds, where $\mathbf{X}$ and $\mathbf{A}$ denote the allowable sets of states and controls, respectively.

The safety filter, operating independently of the behavior-based navigation task, ensures robot safety by aligning with RL agent actions to maintain exploratory integrity during training. It is compatible with any MARL framework as it does not change the RL algorithm as we show in Sec.4.6.4. To ensure the safety filter operates with negligible latency, the underlying NMPC is implemented via acados [74], utilizing the robust multiple shooting formulation described in Sec. 2.3.4.2.

4.5 Experimental Evaluation

The main focus of this work is to achieve behavior-based navigation of mobile robots while ensuring the safety of the robots. We design the experiments so that we can: **(i)** study the effect of using the MPC on the training of the RL agents, showing that we can

achieve collision free training to learn the task in fewer number of episodes (Sec. 4.6.1), (**ii**) compare our approach against baselines in simulations, demonstrating that our approach outperforms state-of-the-art baselines, that our RL agent learns to not rely on the MPC-safety layer indefinitely, and that the MPC-safety layer can be integrated with other RL algorithms (Sec. 4.6.2:4.6.4), (**iii**) show that our approach can be transferred on real robots (Sec. 4.6.5), (**iv**) test the ability of the trained policy to generalize to more agents than used during training (Sec. 4.6.6).

Training Details: For the attention module, we used four attention heads and a hidden dimension of 128. For our approach and the RL baselines, we employed MLPs with two hidden layers of size (256, 256) for both the critics and the actors, using ReLU activation functions. The entropy coefficient in SAC is optimized online. The learning rate for all modules is set to 3×10^{-4}, with a batch size of 128 and a replay buffer size of 1×10^{6}. The observation space and reward are fixed for all RL approaches. In simulations, we used the Turtlebot3 (TB3) robots, while in the real world experiments we used two TB2 robots and one TB4 robot.

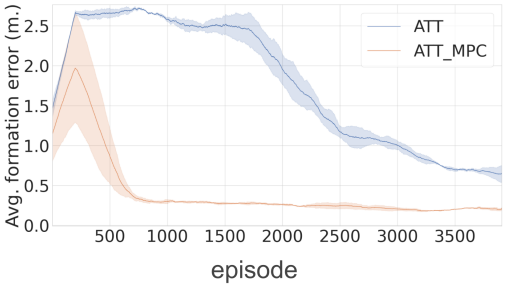
4.6 Experimental Modeling Assumptions

To provide a consistent technical baseline for the multi-agent experimental results discussed in this chapter, we define the underlying modeling assumptions regarding system dynamics, sensing, and safety as follows:

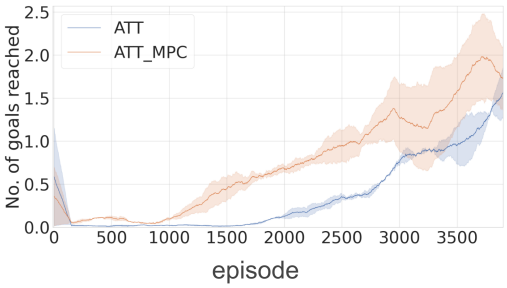
- **System Dynamics**: Every robot in the formation is modeled as a discrete-time kinematic unicycle. The control loop operates at 20 Hz , with actions constrained to a linear velocity $v \in [0, 1.0]\,\mathrm{m/s}$ and an angular velocity $\omega \in [-0.75, 0.75]\,\mathrm{rad/s}$.
- **Observation Space**: Each agent's state includes 40 equiangular LiDAR rays providing a 360° field of view. Similar to the single-agent setup, these 40 readings are obtained by dividing the scan into 40 equidistant sections and extracting the minimum distance from each. The observation also includes the relative coordinates of the formation's centroid and relative information about the two nearest neighbors.
- **Safety Constraint Definitions**: Safety is defined as the elimination of all collisions with static obstacles and neighboring agents during both the training and execution phases. This is enforced through distributed NMPC filters that prioritize obstacle avoidance. Additionally, a sparse collision penalty is included in the reward function to penalize the RL agent for behaviors that lead to intervention by the safety filter.

4.6.1 Training With the MPC Filter

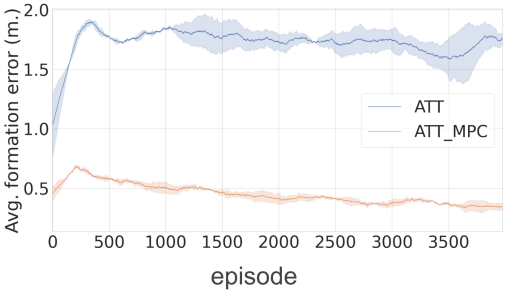
To evaluate the MPC's impact on agent training, we simulated two environments commonly used in the safe MARL literature [90], [91], [104], [106], [107]. The first environ-



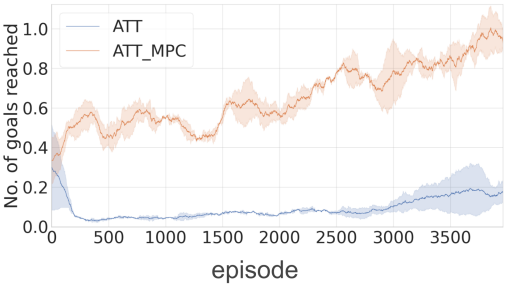
(a) **Average formation error** during the **first** environment.



(b) **Number of goals** reached during the **first** environment.



(c) **Average formation error** during the **second** environment.



(d) **Number of goals** reached during the **second** environment.

Figure 4.5: The figures show the impact of the safety filter on training, with bold lines representing the average and shaded areas indicating the standard deviation across three random seeds. **(a,c)** depict the average formation error in the first and second training environments, while **(b,d)** display the number of goals achieved per episode. The agent with the MPC (**ATT_MPC**) consistently surpasses the pure learning agent (**ATT**) in reducing formation errors and increasing goal achievements.

ment is an empty walled setup to teach the robots to move the centroid to the goal. The second environment includes randomly placed cylindrical obstacles for collision avoidance learning. We train the policies in both environments with and without the MPC. We utilized the same network parameters to isolate the MPC filter's effects. Additionally, we incorporated a penalty in the RL policy for deviations from MPC-safe actions. To maintain stable training despite the MPC's modified actions, we store the RL action in the replay buffer and include the MPC action as the previous action in the observation space. Comparative metrics include average formation error, number of goals reached per episode, and number of collisions. Each episode terminates once the maximum number of steps is reached, or one of the robots collides or is stuck.

The average formation error is defined as follows:

$$Formation\ Error = \frac{1}{N}\sum_{i=1}^{N}[dist_{i,j} - dist_{i,j}^{ref}]$$

where $dist_{i,j}^{ref}$ is the reference distance between robots i and j, while $dist_{i,j}$ is the actual distance.

We denote the agents without the MPC filter as **ATT**, and those with the MPC as

ATT_MPC. The results, displayed in Fig. 4.5, highlight the efficacy of the MPC filter in training enhancement. Notably, **ATT_MPC** reduces the average formation error to less than 0.5 meters within 1,000 episodes, a significant improvement over **ATT**, which requires about 4,000 episodes to achieve similar performance (Fig. 4.5a, 4.5c). This enhancement is largely due to the MPC filter's ability to eliminate collisions, facilitating more effective exploration by the robots.

Additionally, Fig. 4.5b, 4.5d illustrate that including the MPC-filter leads to reaching more goals per episode compared to the pure learning agents. These experiments show that we are able to decrease the number of episodes required to achieve the desired behavior, which is a crucial aspect for safe reinforcement learning, since fewer episodes means fewer resets of the environment in the real-world.

4.6.2 Testing Against Baselines in Simulation

We evaluated our approach against three state-of-the-art baselines to validate its effectiveness in motion planning tasks. The baselines include:

- **MASAC [103]:** A multi-agent variant of the Soft Actor-Critic (SAC) [103], as it outperformed several MARL baselines in the motion planning tasks.
- **MADDPG [113]:** The Multi-Agent Deep Deterministic Policy Gradient [113], a standard benchmark in MARL studies.
- **DMPC [112]:** A decentralized model predictive control approach [112], using the centroid's reference as the goal for each robot, while maintaining inter-robot distances.

The RL agents were trained over two stages, each consisting of 4,000 episodes, with pre-trained agents loaded for further training in the second stage to adapt to obstacle avoiding scenarios. We compare between the agents in different scenarios.

4.6.2.1 Reaching a Single Goal

In the first scenario, we tested the ability of the robots to reach the desired goals starting from different configurations with and without obstacles in the environment. At the start of each episode, the locations of the robots, obstacles, and goal location are randomized. The results are summarized in Tables 4.2 and 4.1. Our approach outperforms the baselines in terms of all three metrics. Additionally, we evaluated our approach in more complex environments by introducing up to eight static obstacles, including cubes and wall sections, as well as dynamic obstacles, while training was limited to cylindrical obstacles. The robots avoided collisions in all cases, but timeouts increased with obstacle density, reflecting the system's conservative behavior and the challenge of navigating constrained spaces. Results are in Table 4.3.

Goals Reached without Obstacles			
Agent	**Success ↑**	**Colls ↓**	**Tout ↓**
Ours	**99.5%**	**0**	**0.5%**
MASAC [103]	58%	8.9%	33.1%
MADDPG [113]	9.5%	57.7%	32.8%
DMPC [112]	13.9%	0	86.1%
Can we execute our method without the MPC?			
Ours without MPC	98.6%	1.4%	0
Integrating the MPC Safety Layer with the Baselines			
MASAC [103] (With MPC)	67.7%	0	32.3%
MADDPG [113] (With MPC)	10.4%	0	89.6%

Table 4.1: Goals reached *without* obstacles (1,000 trials). We compare our method to MASAC, MADDPG, and DMPC, plus ablations (ours w/o MPC) and baselines wrapped with the MPC safety layer. Our approach achieves near-perfect success with zero collisions and minimal timeouts. Removing the MPC filter slightly reduces success and introduces rare collisions. Adding the MPC filter to MASAC eliminates collisions and improves success, while gains for MADDPG are limited. Percentages denote shares over 1,000 trials.

Goals Reached with Obstacles			
Agent	**Success ↑**	**Colls ↓**	**Tout ↓**
Ours	**96.2%**	**0**	3.8%
MASAC [103]	51.8%	26.3%	21.9%
MADDPG [113]	6.6%	81.5%	11.9%
DMPC [112]	6.9%	0	93.1%
Can we execute our method without the MPC?			
Ours without MPC	96%	2.5%	**1.5%**
Integrating the MPC Safety Layer with the Baselines			
MASAC [103] (With MPC)	60.3%	0	39.7%
MADDPG [113] (With MPC)	7.6%	0	92.4%

Table 4.2: Goals reached *with* obstacles (1,000 trials). Our method maintains high success with zero collisions and few timeouts. Disabling the MPC filter lowers success and introduces collisions. Wrapping baselines with the MPC safety layer removes collisions and moderately improves MASAC, while MADDPG benefits little. Percentages denote shares over 1,000 trials.

4.6.2.2 S-Shaped Path Following

In this scenario, we simulated S-shaped paths for the centroid of the robot formation to replicate a real-world application where robots must navigate cooperatively following a path, e.g. conducting surveillance across multiple locations. We defined eight distinct S-shaped paths and initiated the robots from 15 varied starting configurations for each path.

The performance was evaluated based on completion time, collisions, and formation error, with results summarized in Table 4.4. Our approach demonstrated superior per-

No. of Obstacles	Success↑	Timeouts ↓
4	90.3%	9.7%
5	86.7%	13.3%
6	83.8%	16.2%
7	82.3%	17.7%
8	71.4%	28.6%
3 Dynamic Obstacles	81.4%	18.6%
4 Dynamic Obstacles	71.3%	28.7%

Table 4.3: Results for obstacle avoidance tests with varying numbers of obstacles of different shapes, including dynamic ones. While the robots successfully avoid collisions, the number of timeouts increases with obstacle density, reflecting the system's conservative behavior and the challenges of navigating in more constrained environments.

S-Shaped Path				
Agent	**Goals ↑**	**Time [s] ↓**	**Colls ↓**	**Avg form err [m] ↓**
Ours	**100%**	**108.93 ± 14.8**	**0**	0.392 ± 0.17
MASAC [103]	75.8%	134.63 ± 21.3	2.8 ± 3.65	1.25 ± 0.15
MADDPG [113]	12.4%	118.9 ± 28.39	4.16 ± 3.81	4.05 ± 4.06
DMPC [112]	13.4%	162.7 ± 15.53	0	**0.127 ± 0.184**
Can we execute our method without the MPC?				
Ours without MPC	93.7%	112.35 ± 16.7	0.45 ± 1.48	0.42 ± 0.23
Integrating the MPC Safety Layer with the Baselines				
MASAC [103] (With MPC)	79.9%	129.54 ± 38.7	0	0.98 ± 0.18
MADDPG [113] (With MPC)	13.05%	164.9 ± 41.02	0	3.56 ± 3.67

Table 4.4: S-shaped path test (120 trials). We report goal completion rate, time to finish, collisions, and average formation error. Our approach completes all runs fastest with zero collisions. DMPC attains the lowest formation error but has a low completion rate and longer times. Disabling the MPC filter reduces reliability, while adding the MPC safety layer to MASAC removes collisions and yields moderate improvements. Results are aggregated over 120 tests.

formance in completion time and collision avoidance, achieving the least time and zero collisions. While the DMPC showed the smallest formation error, it struggled with efficient positioning around the centroid, resulting in prolonged completion times. Among the learning-based methods, our approach exhibited the lowest formation errors, underscoring its effectiveness in maintaining formation while navigating.

4.6.3 Can we execute our method without the MPC?

In this section, we investigated if the MPC-filter can be disabled after training. We mentioned in Sec.4.6.1 that we added a penalty for deviating from the MPC-safe actions. In Fig. 4.6, we show that this penalty decreases over time, indicating the agents learn to adhere to the safe actions. To further test the agent without the MPC-safety filter, we used an agent previously trained with the MPC-filter, deactivated the MPC-safety filter and tested the agent in the same scenarios. Performance outcomes are detailed in Tables 4.2, 4.1, and 4.4. While the agent without the MPC-filter achieved over 90% in all

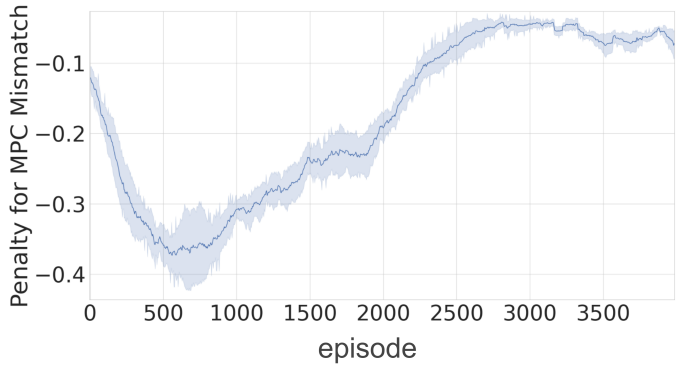


Figure 4.6: The figure shows the penalty given to the agent for deviating from the MPC safe actions.The bold lines show the average while the shaded area show the standard deviation over three different random seeds. The penalty decreases through the training indicating that the agent learns to match the MPC safe actions.

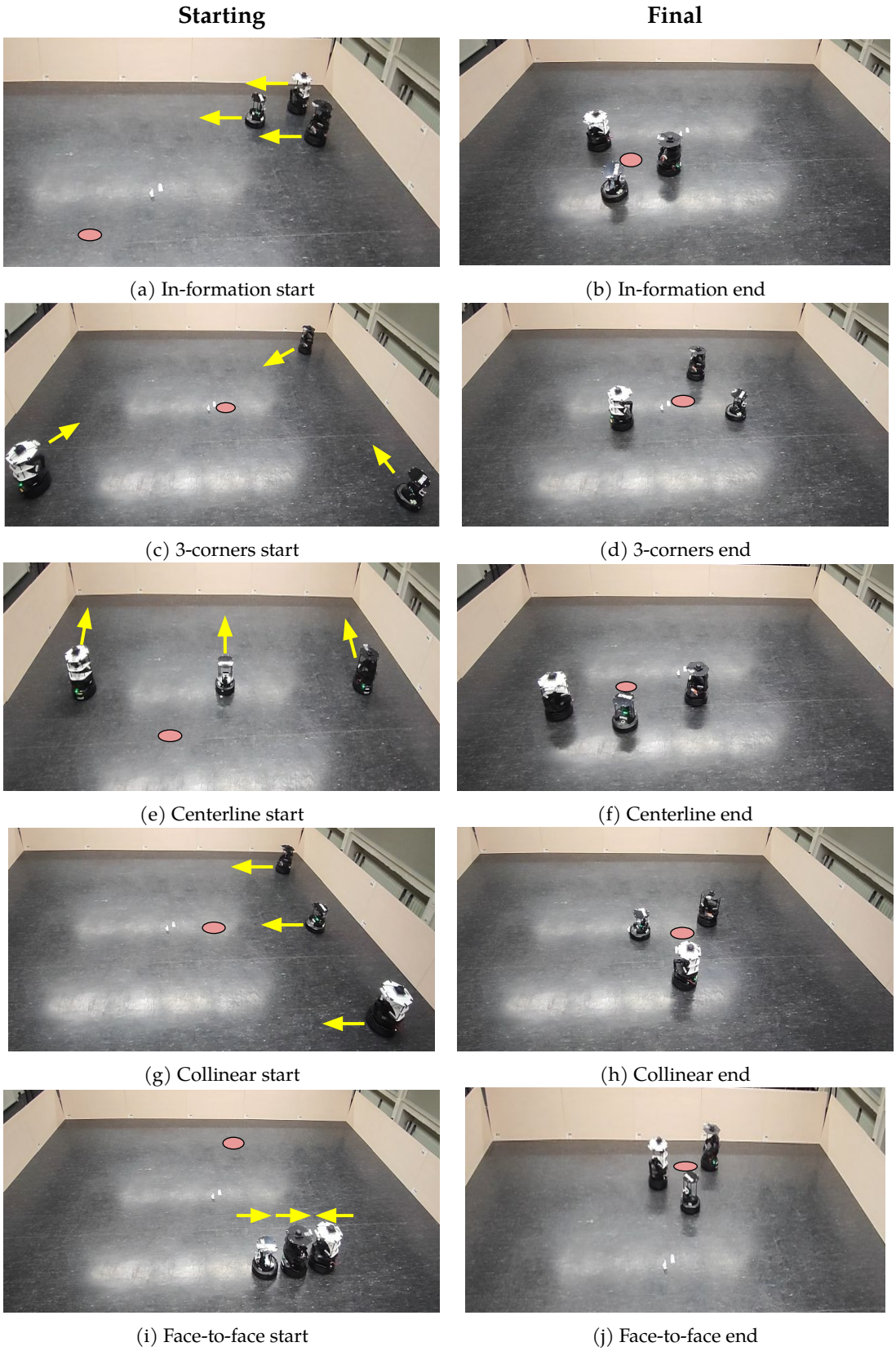
Reaching a Single Goal				
Agent	**Ours**		**MASAC [103]**	
Configuration	**Goals reached ↑**	**Colls ↓**	**Goals reached ↑**	**Colls ↓**
In-formation	100%	**0**	100%	1.0 ± 0
3-corners	100%	**0**	100%	1.0 ± 0
Centerline	100%	**0**	100%	1.5 ± 0.7
Collinear	100%	**0**	100%	0.5 ± 0.7
Face-to-face	100%	**0**	100%	2.5 ± 0.7

Table 4.5: The table shows the performance of our approach against multi-agent SAC (MASAC) in the real-world. Each scenario is tested two times using each approach. Our approach makes zero collisions in all configurations. The last two configurations were previously unseen during the training, nevertheless our approach was able to reach the targets with zero collisions unlike the MASAC.

tests showing that it can be used without the MPC, it failed to avoid collisions as effectively as it did with the MPC enabled, highlighting the MPC's critical role in ensuring safety.

4.6.4 Integrating the MPC Safety Layer with the Baselines:

We evaluated the baseline models after being retrained with the MPC safety layer and evaluated their performance using the established tests. The outcomes are detailed in Tables 4.2, 4.1, and 4.4. Incorporating the MPC with MASAC eliminated collisions, which increased the number of goals achieved but also led to more timeouts, indicating the MPC's intervention to prevent collisions. Adding the MPC to MADDPG did not yield improvements, potentially due to the inherent stability issues of DDPG [75]. The frequent timeouts suggest that the MPC often stops the robots to avoid collisions, resulting in timeouts due to the robots being stuck.



(a) In-formation start (b) In-formation end

(c) 3-corners start (d) 3-corners end

(e) Centerline start (f) Centerline end

(g) Collinear start (h) Collinear end

(i) Face-to-face start (j) Face-to-face end

Figure 4.7: Target Reaching Configurations test. The top headers indicate starting vs. final configurations; each row shows one scenario. Yellow arrows indicate starting orientation; red circles mark the target centroid. Configurations are ordered top-to-bottom by difficulty. The **Collinear** and **Face-to-face** configurations were unseen during training; our approach still succeeds with zero collisions.

4.6.5 Real-World Experiments

In the accompanying video, we demonstrated the behavior of initial (random) policies with and without the MPC filter on real robots, showcasing that our approach enables safe training on physical robots. However, training multiple robots in the real world is currently impractical due to time requirements. Our framework, although collision-free, requires 25 hours of simulation time, which translates to about 100 hours of real-world training.

Previous studies on safe MARL [90], [91], [92], [104], [106] have only evaluated final policies in simulations. In contrast, we validate our trained policies on real robots. The aim of these experiments is to demonstrate the zero-shot transfer of behavior-based navigation and the safety of our approach.

For comparison, we selected MASAC as a baseline due to its superior simulation performance over MADDPG and DMPC (Sec. 4.6.2), and trained both for the same number of episodes. The policies were then tested in real-world scenarios identical to those used in simulations.

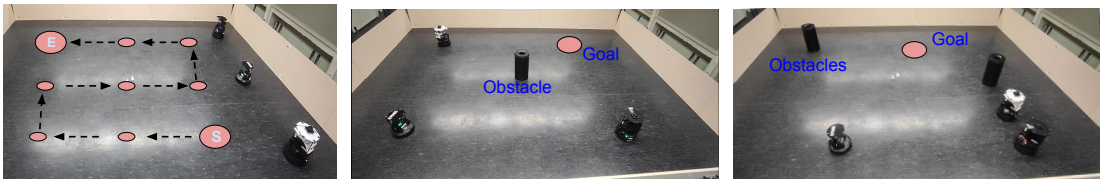
4.6.5.1 Reaching a Single Goal

We tested five different starting configurations with a target location for the formation centroid, as shown in Fig. 4.7. Each configuration was tested twice for each policy, with results summarized in Table 4.5. Our approach consistently achieved zero collisions across all configurations, including unseen scenarios, demonstrating superior generalization and safety. Additionally, our method reached the targets faster than MASAC, as it better coordinates robot movements.

In unseen scenarios, our approach required more maneuvers to avoid collisions, unlike MASAC, where robots reached the targets but failed to avoid collisions. For example, in the **face-to-face** configuration, our robots rotated and moved sequentially to avoid collisions, whereas MASAC robots collided and moved while touching, which explains the less time taken (these results are shown in the video). All tested configurations were reachable by both methods. We further tested more configurations that were not reachable by the baseline in the video. Both approaches were tested in obstacle avoidance scenarios. Fig. 4.8 show two of these scenarios, and the results are shown in Tables 4.7.

4.6.5.2 S-Shaped Path Following

In the S-shaped path scenario, Fig. 4.8a, we run the test three times for each approach and recorded the number of collisions, the average formation error over the whole path, number of timeouts, as well as the time taken to complete the path. The results are summarized in Table 4.6. Our approach was able to successfully reach all targets without making collisions, and in less time compared to the MASAC. Additionally, our approach had less average formation error compared to the MASAC over all the runs.



(a) S-Shaped path (b) Scenario A (c) Scenario B

Figure 4.8: The figures show the S-Shaped path tests and two scenarios of the obstacle avoidance tests. (**a**) shows the S-shaped path. The red circles indicate the target locations for the centroid of the formation, where the first and final targets are indicated by the letters **S**, **E**, respectively. (**b,c**) show two different scenarios for obstacle avoidance. The robots start from the shown configurations and navigate towards the goal while avoiding collisions with the obstacles shown in the figures.

S-Shaped Path				
Agent	**Goals ↑**	**Time [s] ↓**	**Colls ↓**	**Avg form err [m] ↓**
Ours	**100%**	**143.2 ± 5.7**	**0**	**0.34 ± 0.016**
MASAC [103]	96.2%	212.2 ± 53.8	7.3 ± 1.52	0.58 ± 0.35

Table 4.6: S-shaped path results (three runs per approach). MASAC reached 96.2% of targets because some runs exceeded the step limit. During testing, if robots take more than 200 steps to reach a target, the next target is issued and the previous one is counted as a timeout. Our approach records zero collisions.

Goals Reaching With Obstacles				
Agent	**Success ↑**	**Tout ↓**	**Colls ↓**	**Touts ↓**
Ours	**100%**	0	**0**	**0**
MASAC [103]	0%	0	100%	0

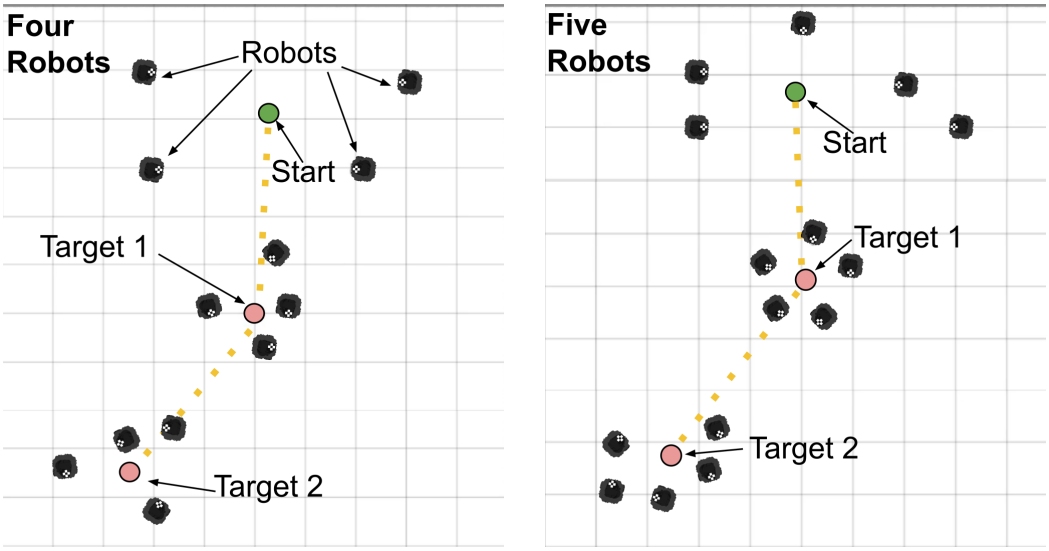
Table 4.7: Obstacle-avoidance goal-reaching results. Our approach completes all four tests with zero collisions. MASAC is unsuccessful at reaching goals in these tests due to collisions with obstacles.

Goals Reaching Test With More Robots			
Number of Robots	**Goals reached**	**Collisions**	**Timeouts**
3	99%	0	1%
4	97%	0	3%
5	91%	0	9%
6	79%	0	21%
7	76%	0	24%
8	70%	0	30%

Table 4.8: The table shows the performance of our approach when generalizing to more robots, over 100 episodes. The percentage reflects the proportion of achieved goals, collisions, and timeouts in relation to the total number of trials. Our approach is able to generalize to more robots with zero collisions. However, as the number of robots increased, we observed an increase in timeouts, attributed to the conservative nature of the MPC filter.

4.6.6 Generalizing to More Robots

Finally, we show that by relying only on information from two neighbors, our method scales to more robots while maintaining a fixed observation size, regardless of the num-



(a) Testing with four agents

(b) Testing with five agents

Figure 4.9: Example scenarios (**a**) and (**b**) show the generalization of the trained policy to a larger team of robots. The robots successfully move their centroid from the starting location (green circle) to the target location (red circle). The yellow lines indicate the movement of the centroid through the different locations.

ber of robots.

We evaluated the performance of the trained policy with up to eight robots, as shown in Fig. 4.9. For each robot, we defined two neighbors and tested the policy using the previously introduced goal-reaching scenarios. Metrics included the number of goals reached, collisions, and timeouts, with results summarized in Table 4.8. The robots successfully cooperated to move their centroid to the target locations, with zero collisions. However, as the number of robots increased, we observed an increase in timeouts, attributed to the conservative nature of the MPC filter and out-of-distribution scenarios introduced by additional robots. The policy was trained in an environment with only three robots and static obstacles, without exposure to dynamic obstacles. With more robots, they act as dynamic obstacles, leading to local congestion where robots block each other's paths, ultimately resulting in timeouts. These trade-offs are expected in safety-critical systems, where eliminating collisions is paramount. Addressing these limitations by incorporating dynamic obstacle scenarios into the training, along with integrating an oracle model to predict the future locations of nearby moving agents, represents a promising direction for future work to enhance the system's robustness and scalability.

4.7 Conclusion

In this study, we presented a safe multi-agent reinforcement learning (MARL) approach to achieve behavior-based cooperative navigation without individual reference targets in real-world scenarios. Our findings demonstrate that relying on the centroid of the

formation is sufficient for robots to navigate cooperatively. By integrating MARL, and model predictive control (MPC)-based safety filters, we ensured zero collisions during training and achieved faster convergence. The inclusion of a safety layer not only eliminated collisions but also significantly improved training efficiency, leading to faster convergence of the MARL policies. This has crucial implications for addressing the sim-to-real gap, as it highlights the benefits of incorporating safety layers into RL training. Our trained policies, applied to real robots, outperformed baseline methods in various tests in terms of success rate, number of collisions and completion times, confirming the effectiveness of our approach. Our experiments show that training on real robots is safe, eliminating collisions even during early exploration stages. These results collectively demonstrate that distributed MPC-based filters provide a scalable and robust solution for safe behavior-based formation control, successfully addressing the coordination and safety requirements defined in RQ2.

5 A Dynamic Safety Shield for Safe and Efficient Reinforcement Learning of Navigation Tasks

Abstract

To overcome the exploratory limitations of rigid safety overrides discussed in the previous chapter, this chapter introduces an adaptive shielding framework. This approach allows the RL agent to dynamically tune the parameters of the underlying safety shield, optimizing the balance between strict safety enforcement and task performance. Reinforcement learning (RL) has been successfully applied to a variety of robotics applications, where it outperforms classical methods. However, the safety aspect of RL and the transfer to the real world remain an open challenge. A prominent field for tackling this challenge and ensuring the safety of the agents during training and execution is safe reinforcement learning. Safe RL can be achieved through constrained RL and safe exploration approaches. The former learns the safety constraints over the course of training to achieve a safe behavior by the end of training, at the cost of high number of collisions at earlier stages of the training. The latter offers robust safety by enforcing the safety constraints as hard constraints, which prevents collisions but hinders the exploration of the RL agent, resulting in lower rewards and poor performance. To overcome those drawbacks, we propose a novel safety shield, that combines the robustness of the optimization-based controllers with the long prediction capabilities of the RL agents, allowing the RL agent to adaptively tune the parameters of the controller. Our approach is able to improve the exploration of the RL agents for navigation tasks, while minimizing the number of collisions. Experiments in simulation show that our approach outperforms state-of-the-art baselines in the reached goals-to-collisions ratio in different challenging environments. The goals-to-collisions ratio metrics emphasizes the importance of minimizing the number of collisions, while learning to accomplish the task. Our approach achieves a higher number of reached goals compared to the classic safety shields and fewer collisions compared to constrained RL approaches. Finally, we demonstrate the performance of the proposed method in a real-world experiment.

5.1 Introduction

In the last decade, reinforcement learning (RL) has been shown to outperform classical methods in navigation tasks [114], [115]. This is mainly due to the fact that RL approaches are able to directly map sensory information into actions, enabling them to learn in complex scenarios that would otherwise necessitate lots of engineering efforts. Nevertheless, the safety aspect of RL and the transfer of the trained policies to the real-world remain challenging.

Safe RL has been proposed to ensure the safety of the RL agents during training and execution and two main directions have emerged in the field. First, constrained

reinforcement learning: this approach formulates the RL as a constrained Markov decision problem to minimize safety constraints violations during training [39], [40], [116], [117]. This method trains an additional network to estimate the cost of constraint violations. The policy is then trained to maximize rewards while minimizing those costs. Although these methods often achieve safe behavior by the end of training, they tend to violate constraints frequently during early learning. Additionally, the end-to-end nature of these policies makes their transfer to the real-world even more challenging due to the sim-to-real gap.

The second approach to achieve safe RL is safe exploration. In addition to the RL agent responsible for achieving the task (task agent), a safe policy, also referred to as a safety shield, is introduced to ensure the safety of the task agent during training and execution [110], [118], [119]. The safety shield utilizes knowledge about the dynamics of the robot, without having knowledge about the dynamics of the environment, to override unsafe actions from the task agent during training and execution. Although, this approach is more reliable and suffers from less constraints violations, it tends to overly restrict the exploration process, which consequently leads to less rewards achieved by the RL agents. This is caused by the formulation of the safety shields, which focus mainly on collisions avoidance and perform with limited prediction horizons.

To address these shortcomings, we propose integrating the long-horizon predictive capabilities of reinforcement learning with the robustness of optimization-based controllers, see Fig. 5.1. Specifically, we combine a model predictive control (MPC)-based safety shield with an RL agent to provide enhanced guidance to the task agent. This agent is referred to as the supervisor agent and is responsible for dynamically adjusting the weights of the constraints based on the current observations. Additionally, to ensure that the safety shield does not overly constrain the task agent's exploration, the supervisor agent also adjusts the weights to align the shield's actions with those of the task agent. The supervisor agent is responsible for avoiding collisions while matching the actions of the task agent, without having knowledge about the main task of the task agent. A clear advantage of not informing the supervisor agent about the main task is to minimize the exploration needed for the supervisor agent to converge. This results in a small number of constraints violations, while not affecting the exploration of the task agent, as we show in the experiments.

In this work, we focus specifically on safety in navigation tasks, where the task agent controls the robot's linear and angular velocities. Thus, the considered constraints are related to obstacle avoidance and constraints violations are collisions done by the learning robot.

In summary, the main contributions of our work are:

(i) A novel safety shield for navigation tasks that combines the robustness of the MPC shields with the long-horizon capabilities of the RL agents, by allowing an additional RL agent to dynamically adjust the weights of the constraints and the weights for matching the task agent's actions.

(**ii**) A supervisor RL agent, without access to goal-related information, to dynamically adjust the MPC shield online. This goal-independent training results in small number of collisions as we show in the ablation study.

(**iii**) We show over several simulation environments of increasing difficulties the superiority of our approach over several state-of-the-art baselines w.r.t. the reached goals-to-collisions ratio. Finally, we demonstrate the performance of our approach in real-world scenarios.

5.2 Related Work

5.2.1 Constrained Reinforcement Learning

Several studies proposed formulating the RL task as a constrained Markov decision problem, to take the constraints into consideration during training. In [40] the authors modified the trust-region policy optimization (TRPO) method to include the constraints on expected costs in the update rule of the algorithm. The study [120] introduced the Lagrangian TRPO and Lagrangian proximal policy optimization methods, and showed that constrained RL methods achieve fewer constraints violations compared to the unconstrained RL at the cost of reduced rewards. However, a common issue in the Lagrangian constrained RL methods was instability during training. In [39] the authors addressed this instability by introducing PID control over the Lagrangian update and showed improved performance. [121] suggested pre-training a separate recovery policy along with a critic that learns the constraints violations offline then continues the training online. The pre-training phase requires a replay buffer that demonstrates examples of constraints violations. [122] proposed using a safety budget, which represents the remaining allowable constrains violation, instead of the cost to reduce constraints violations. [117] combined model-based constrained RL with the recovery policy from [121] and showed reduction in the constraint violation compared to several model-free constrained RL baselines. These methods learn the constraints during training to decrease the rate of constraints violation over the course of training, which results in high number of constraints violations during the initial training stages.

5.2.2 Safe Exploration

On the other spectrum of safe RL lie safe exploration approaches, which promote safer exploration by eliminating actions that would lead to unsafe states. [37] proposed using MPC-based safety shield to override unsafe actions from the RL agent, where the safe states were formulated as hard constraints. [118] pre-trained a neural network to act as the safety shield, and added it to the policy network that is trained from scratch. [123] used a state estimator in addition to the safety shield to improve the agent safety. [106] utilized predictive safety shields to ensure the safety of the agents in multi-agents

scenarios. [119] used control barrier functions [105] as a shielding mechanism to ensure the safety of the agents. Different from these methods, we take advantage of the RL long-horizon prediction capabilities to tune the safety shield used in training the RL task agent. This leads to a less constrained shield that does not hinder the exploration of the task agent as we show in the experiments. Moreover, softening the hard constraints and tuning their weights online leads to less failures of the MPC solver, where the MPC is unable to find a solution, which commonly happens when considering several constraints as in [34], [38].

5.2.3 Positioning relative to Constrained RL and Safe Exploration

To clarify the positioning of our adaptive safety shield, we distinguish it from existing safe RL formulations:

While constrained RL methods *internalize* the cost through dual optimization, they often face the problem of conflicting gradients. This forces the agent to navigate a trade-off between task progress and safety, which frequently leads to training instabilities and high violation rates during the early stages of learning as the multipliers adapt.

On the other hand, safe exploration approaches *externalize* safety by using rigid shields to override unsafe actions. While these offer robust protection, they often hinder exploration by enforcing hard constraints that may be over-conservative, potentially leading to suboptimal task performance or frequent solver failures in complex environments.

In contrast, our approach occupies a middle ground by decoupling these objectives into a hierarchical dual-agent structure. By utilizing an RL supervisor to dynamically tune the cost parameters of an optimization-based MPC shield, we maintain the robustness of MPC while leveraging the long-horizon predictive power of RL to prevent over-conservatism. This ensures safe exploration from the start of training without the need for delicate manual weight tuning or the instability of dual-variable optimization.

5.3 Our Approach

In this work, we propose a novel safety shield that is tuned online using an RL agent, i.e., the supervisor agent, to ensure the safety of the task agent, see Fig.5.1. The shield along with the supervisor agent do not have access to goal-related information, which is handled by the task agent. Our aim is to develop a safety shield that minimizes the number of collisions, while not hindering the exploration process of the task agent.

First, we introduce the considered navigation task. We then explain the dynamic safety shield and the differences from the classic MPC-safety shield [37], [38]. Afterwards, we introduce the supervisor RL agent responsible for adjusting the shield online. Finally, we introduce the task agent used for solving the task.

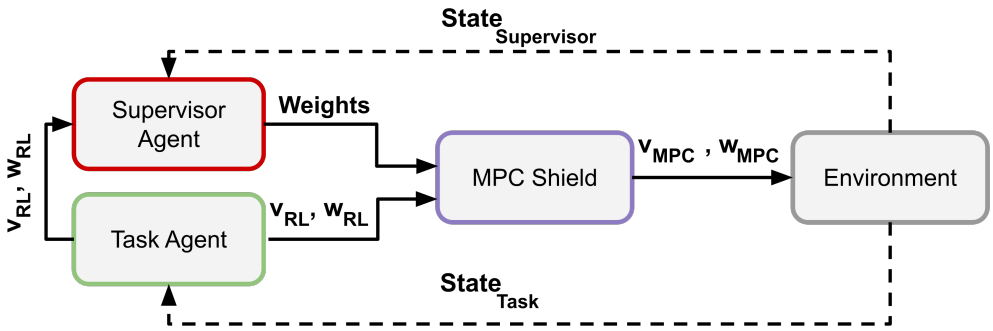


Figure 5.1: Architecture of our approach. The task agent (green) is responsible for learning the navigation task. The agent receives the $State_{Task}$ from the environment and outputs the linear and angular velocities (v_{RL}, w_{RL}). The supervisor agent (red) receives the $State_{Supervisor}$ from the environment and outputs the *Weights* for aligning the MPC-shield's actions with the task agent's actions, and the weights of the constraints. The MPC shield solves the optimal control problem (Eq. 5.3) using the all mentioned weights to find the safe actions v_{MPC}, w_{MPC}.

5.3.1 Navigation Task

We focus on the navigation task, where the RL task agent should learn how to navigate to goals, as fast as possible, without collisions in unknown environments. We assume that the robot is equipped with a lidar sensor and is given the relative distance and heading to the goal. The lidar sensor provides 100 equidistant beams over a range of $360°$, giving obstacle information around the robot. The task agent uses this information to produce the linear and angular velocities for the robot.

5.3.2 Dynamic Model Predictive Control Safety Shield

The MPC shield requires a mathematical model of the robot to predict the future states based on the current state and the calculated control sequence. An optimization control problem is solved at each time step to find the control sequence that minimizes a predefined cost function. Only the first control action in the control sequence is applied to the robot. It is important to note that the mathematical model represents the kinematics of the robot only and does not include the dynamics of the world model as in model-based RL [117], [124].

Prediction Model: The discrete-time model of the robot is:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} \cos\theta_t & 0 \\ \sin\theta_t & 0 \\ 0 & 1 \end{bmatrix} \mathbf{a}_t \, \Delta t. \tag{5.1}$$

where $\mathbf{x} = [x, y, \theta]^T$ represents the state of the robot, which is the position and heading relative to the starting position. $\mathbf{a} = [v, \omega]^T$ is the action, which is the linear and angular velocities of the robot.

Optimal Control Problem (OCP): To ensure the safety of the robot while minimizing the intervention of the safety filter, the optimization problem at time step t is formu-

lated as:

$$\min_{\substack{\mathbf{x}_{t:t+T|t},\\ \mathbf{a}_{t:t+T-1|t}}} \|\mathbf{a}_{RL} - \mathbf{a}_{t|t}\|^2_{R_0} + \sum_{k=1}^{T-1} \|\mathbf{a}_{t+k|t}\|^2_R \tag{5.2a}$$

$$\text{s.t.} \quad \mathbf{x}_{t|t} = \mathbf{x}_t, \tag{5.2b}$$

$$\mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{a}_{t+k|t}), \forall k = 0, 1, ..., T-1 \tag{5.2c}$$

$$\mathbf{a}_{t+k|t} \in \mathbf{A}, k = 0, 1, ..., T-1 \tag{5.2d}$$

$$\mathbf{x}_{t+k|t} \in \mathbf{X}, \forall k = 0, 1, ..., T-1, \tag{5.2e}$$

$$dist_{obst}^{t+k|t} > \delta, \forall obst = 1, 2..., M, \forall k = 0, 1, ..., T-1, \tag{5.2f}$$

where T represents the prediction horizon. The weighted Euclidean norm, represented by $\|\cdot\|^2_R$, is defined as $\|\mathbf{x}\|^2_R = \mathbf{x}^\mathsf{T} R\mathbf{x}$, where R is a positive definite weighting matrix. The notation $t+k|t$ indicates predictions at time $t+k$, assuming the current time is t. The first term in (5.2a) measures the deviation between the task agent's proposed action $\mathbf{a}_{RL}$ (Action$_{RL}$) and the initial MPC action $\mathbf{a}_t$ (Action$_{MPC}$). The second term minimizes the magnitude of future control signals $\mathbf{a}_{t+k}$, promoting smoother transitions. Weight matrices R_0, R are used to optimize the performance of the shield and are manually tuned. Constraints (5.2b–5.2f) are hard constraints: (5.2b) ensures the initial state of the model matches the actual state of the robot, (5.2c) enforces the robot's dynamic model, (5.2d) and (5.2e) bounds the actions and states within the predefined limits, where $\mathbf{X}$ and $\mathbf{A}$ denote the allowable sets of states and controls, respectively. (5.2e) enforces the distance between the robot and the nearest M obstacles to be larger than a safety threshold distance δ. M is a design parameter representing the maximum number of obstacles considered by the MPC. Since the MPC process raw lidar data directly, the lidar data is divided into M equal sectors and only the beam with the smallest distance in each sector is considered as obstacle. In this work, we use $M = 4$.

The main drawbacks of the above formulation are that considering several obstacles as hard constraints in the MPC problem can often lead the solver to fail to find a feasible solution online as in [34], and manually tuning the weight matrix R_0 to achieve a reasonable performance is a time-consuming process. That is why we propose modifying the OCP as follows:

$$\min_{\substack{\mathbf{x}_{t:t+T|t},\\ \mathbf{a}_{t:t+T-1|t}}} \|\mathbf{a}_{RL} - \mathbf{a}_{t|t}\|^2_{R_0} + \sum_{k=1}^{T-1} \|\mathbf{a}_{t+k|t}\|^2_R + \sum_{obst=1}^{M} \frac{\omega_{obst}}{dist_{obst}^{t+k|t}} \tag{5.3a}$$

$$\text{s.t.} \quad \mathbf{x}_{t|t} = \mathbf{x}_t, \tag{5.3b}$$

$$\mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{a}_{t+k|t}), \forall k = 0, 1, ..., T-1 \tag{5.3c}$$

$$\mathbf{a}_{t+k|t} \in \mathbf{A}, k = 0, 1, ..., T-1 \tag{5.3d}$$

$$\mathbf{x}_{t+k|t} \in \mathbf{X}, \forall k = 0, 1, ..., T-1, \tag{5.3e}$$

where the green terms in the cost function 5.3a indicates our modifications. The weight

matrix R_0 is now tuned online, while the obstacle avoidance terms scaled by the weights ω_{obst} are now added to the cost function. However, setting these weights equally often leads the robot to being stuck when surrounded by obstacles [38]. That is why we learn these weights online based on the observations of the supervisor agent which we discuss in the next section. We do not tune the R matrix online, We do not tune the R matrix online, as its purpose is to reduce the control effort over the rest of the prediction horizon.

5.3.3 Supervisor Reinforcement Learning Agent

To adjust the MPC shield online, we use a soft-actor-critic [75] (SAC) agent, which we call the supervisor agent. The role of the supervisor agent is to modify the weights of the obstacle terms used in the cost function of the MPC shield 5.3a, in addition to adjusting the weights for matching the actions from the task agent, see Fig. 5.1. We adapt the Markov decision process (MDP), which is described by a tuple M: (S, A, R, P, γ). Where S is the set of states, A is the set of actions, $R(s, a)$ is the reward function, $P(s'|s, a)$ is the transition probability, and γ is the discount factor. An agent in state $s \in S$ takes an action $a \in A$ resulting in the next state $s' \in S$, which is rewarded by reward r and discounted by factor γ. The action a is chosen according to a policy π that determines for each state which action the agent will take. The transition from state s to state s' upon taking action a is determined by the transition probability P.

The **observation space** for the supervisor agent includes the lidar data, the action from the task agent, the previous action from the MPC shield, and the relative angles and distances of the closest M obstacles fed to the MPC, which empirically improves performance. The **action space** of the agent includes the two weights for penalizing the deviation between the task agent and MPC shield, in addition to the M-dimensional weights corresponding to the obstacles considered by the MPC. We formulate the **reward function** as follows:

$$r = \begin{cases} -r_{collision} & \text{if collision or stuck,} \\ -min\_dist_{obst} \cdot \|Action_{RL} - Action_{MPC}\|^2 & \text{otherwise} \end{cases} \tag{5.4}$$

Since the supervisor agent is not responsible for the task completion, we do not give rewards for reaching the goal. Instead, we give the agent a large penalty $r_{collision}$ if the robot collides or if the robot is not moving for several consecutive steps (stuck), or a continuous penalty as a function of the distance to the single nearest obstacle and the difference between the action of the task agent and the MPC shield. That is, the further the robot is from the obstacles, the higher the penalty is for not matching the action from the task agent. As the robot approaches an obstacle, the penalty decreases, prioritizing safety over action alignment. This encourages the supervisor agent to follow the task agent's actions when far from obstacles and adjust them near obstacles to prevent collisions.

5.3.4 Task Agent

The task agent is also a SAC agent, whose observation space includes the lidar data, the previously taken action by the robot (v_{MPC} and ω_{MPC}), and the relative angle and distance to the goal location. The action space consists of v_{RL} and ω_{RL}. The reward function is as follows:

$$r = \begin{cases} r_{goal} & \text{if goal is reached,} \\ (goal\_dist_{t-1} - goal\_dist_t) & \text{otherwise.} \end{cases} \tag{5.5}$$

where the agent receives a large reward r_{goal} for reaching the goal and a continuous term that rewards the progress towards the goal. We do not penalize the collisions for this agent and rely on the supervisor instead to eliminate them.

5.3.5 Hierarchical Training Details

The hierarchical architecture is trained using two separate Soft Actor-Critic (SAC) instances. To achieve stable interaction between the task agent and the adaptive safety shield, we employ the following training protocol:

1. **Decoupled Optimization**: The task agent and supervisor are optimized independently with separate replay buffers. This prevents conflicting gradients, allowing the task agent to focus solely on goal-reaching while the supervisor focuses on safety and action alignment.

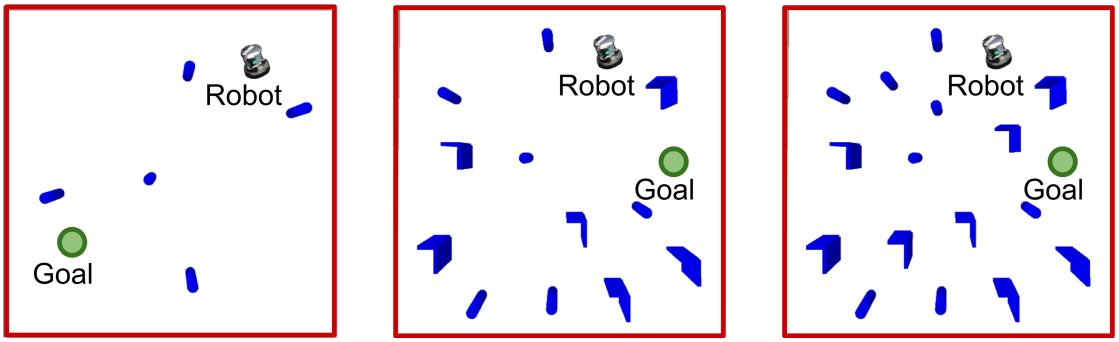
2. **High-Risk Sample Repetition**: To address the sparsity of safety violations, any transition resulting in a collision or a "stuck" state is added to the supervisor's replay buffer three times. This over-sampling ensures the supervisor maintains a high-fidelity representation of dangerous states.

3. **Asynchronous Update Frequency**: The supervisor agent is trained with a higher update-to-step ratio than the task agent. This ensures the safety shield remains responsive and adapts its parameters rapidly as the task agent explores new regions.

4. **Learning Rate Annealing**: A linear decay is applied to the supervisor's learning rate throughout the training process. This stabilizes the safety shield's parameter tuning in the final stages of learning.

5.4 Experiments

This work focuses on achieving safe reinforcement learning in navigation tasks by minimizing collisions without restricting the task agent's exploration. The experiments are designed to: (1) evaluate the impact of the dynamic shield on RL training, (2) compare the proposed approach with baseline methods in simulations, (3) analyze the effect of



(a) Environment 1 (b) Environment 2 (c) Environment 3

Figure 5.2: Environments used in the experiments, Fig.**a** environment with five pillars (blue), and Fig.**b** environment, which contains six pillars and six L-shaped walls (blue). Fig.**c** environment with eight pillars and eight L-shaped walls. All the obstacles are placed randomly at the beginning of each episode.

excluding task-dependent information for the supervisor, and (4) illustrate how the dynamic shield adjusts the constraints' weights on the real robot.

5.4.1 Baselines

For a fair comparison, we chose the baselines such that all of them are model-free RL, all baselines are trained from scratch and do not require pre-training, the baselines include variants of constrained-RL and safe exploration methods: **(i) SAC [75]**: Unconstrained SAC. **(ii) SAC-Lagrangian [120] (SAC_LAG):** The Lagrangian variant of the SAC is a constrained-RL approach which uses two additional critic networks (Q_{cost}) to estimate the costs and optimizes the Lagrange multiplier online to balance the goal reaching with the rate of collisions. **(iii) SAC-PID [39] (SAC_PID)**[3]**:** A PID controller [126] is used along with the SAC-Lagrangian to update the Lagrange multiplier. The method has been shown to stabilize the training of the costs' critic (Q_{cost}) and has been commonly used as a baseline in safe RL studies. **(iv) MPC Safety Shield [38] (MPC_Tuned):** The MPC-based safety shield is a pre-tuned safety shield that ensures zero collisions in navigation scenarios. It has already been used in safe multi-agent RL and is classified as a safe exploration method.

5.4.2 Experimental Setup

We carry out the experiments in PyBullet [127] in three different environments. The first environment, Fig.5.2a, contains five cylindrical obstacles (pillars) placed randomly in the environment at the beginning of each episode, the second environment consists of six pillars and six L-shaped walls, Fig.5.2b, while the third consists of eight pillars and eight L-shaped walls, Fig.5.2c. We chose the pillars as they are commonly used in obstacle avoidance scenarios, and the L-shaped walls as they represent challenging local minima during trajectory planning. During training, the RL agents, the obstacles

[3]We used the same code as in [125] for SAC_LAG and SAC_PID

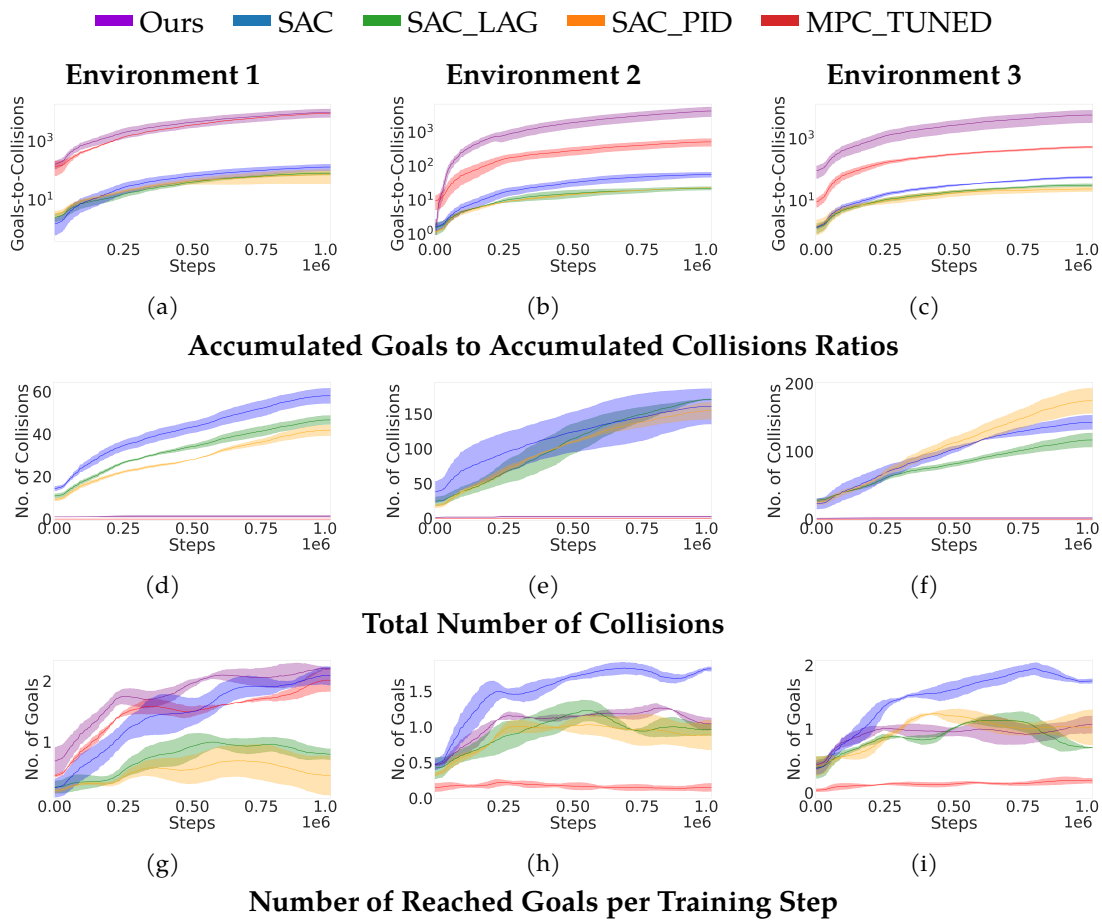


Figure 5.3: Results for all approaches in the three environments. Bold lines show the average of three random seeds; shaded areas show the standard deviation. Our approach achieves the highest goals-to-collisions ratio and near-zero collisions, overlapping with MPC_TUNED at zero (second row).

locations, goals, and the robot location are randomized at the beginning of each episode. We apply episodic training, where the episode terminates if the robot collides or if it is not moving for 30 consecutive steps, or the maximum number of steps is reached. The robot is capable of reaching multiple goals within a single episode; a new goal is randomly generated each time the previous one is reached.

All the agents have been trained for one million steps from scratch. To assess the performance of the agents, we use three metrics as follows: **(i) Accumulated Number of Goals-to-Accumulated Number of Collisions Ratio:** This metric has been introduced in [121] to assess the performance of safe RL approaches as it signifies the importance of achieving more goals and less collisions. A higher ratio indicates fewer collisions while achieving more goals, which is desirable. In case of zero collisions, we set the number of collisions to one, to avoid division by zero. **(ii) Accumulated Number of Collisions:** Additionally, we count the total number of collisions during the training, as this is the main focus of safe RL. Fewer collisions indicate better safety adherence. **(iii) Number of Reached Goals:** Finally, we keep track of the number of goals reached at

each training step to indicate the progress of the task agent. A higher number indicates better behavior from the task agent.

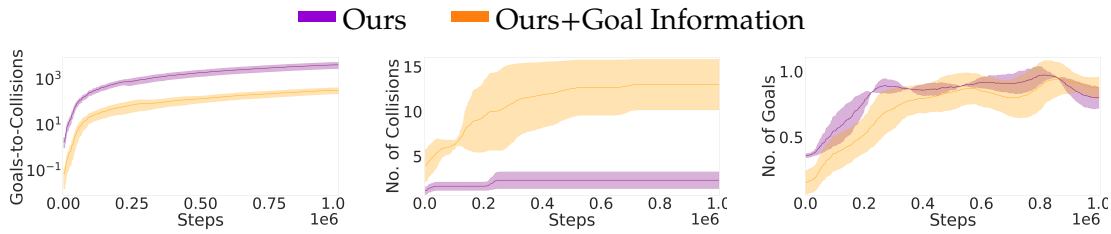
5.5 Experimental Modeling Assumptions

Consistent with the requirement for technical detail across chapters, we define the following assumptions:

- **System Dynamics**: The robot is modeled as a discrete-time kinematic unicycle. For this specific framework, the control frequency is set to 5 Hz ($\Delta t = 0.2\,\mathrm{s}$) for both the agents and the MPC shield. Physical action limits are set to $v \in [0, 1.0]\,\mathrm{m/s}$ and $\omega \in [-0.75, 0.75]\,\mathrm{rad/s}$.
- **Observation Space**: The task agent utilizes 20 equiangular LiDAR rays, calculated by taking the minimum distance of 20 equidistant sections of the scan. The supervisor's observation includes these rays, the task agent's proposed action, and relative data (distance and angle) for the M closest obstacles.
- **Safety Constraint Definitions**: Safety is defined as the absolute avoidance of collisions with static obstacles. This is enforced through the adaptive safety shield, which dynamically modulates the weight of the obstacle terms in the MPC cost function to balance collision avoidance with exploratory freedom.

5.5.1 Results

Figure 5.3 shows the results of the three metrics for all the methods in the three environments. The figures show the means (bold lines) and standard deviations (shaded areas) over three runs for each approach. Figures 5.3a, 5.3b, and 5.3c illustrate that our approach achieves the highest goals-to-collisions ratio in all the environments compared to the baselines. In the first environment, MPC_TUNED has a high ratio as the MPC shield is able to eliminate the collisions completely while navigation to the goals. In the more challenging environments, the MPC_TUNED still maintains zero collisions, see Fig. 5.3d, 5.3e, and 5.3f but at the cost of over constraining the task agent resulting in the least number of goals reached among all the baselines. The unconstrained SAC is able to reach the highest number of goals, Fig. 5.3g, 5.3h, and 5.3i in all environments at the cost of large number of collisions, resulting in low accumulated goals-to accumulated collisions ratios. The constrained RL approaches SAC_LAG and SAC_PID on the other hand are able to decrease the number of collisions compared to the SAC at the cost of less goals reached, consequently leading to low ratios as well. These results demonstrate our method's ability to minimize collisions without over constraining the task agent.



(a) Goals to Collisions Ratio (b) Total Number of Collisions (c) Number of Reached Goals

Figure 5.4: Results for the ablation study over three random seeds. Introducing goal information to the supervisor agent results in more collisions as it explores to reach more goals. The goals-to-collisions ratio without goal information is higher, indicating that adding goal information to the supervisor does not improve task-agent performance.

5.5.2 Ablation Study

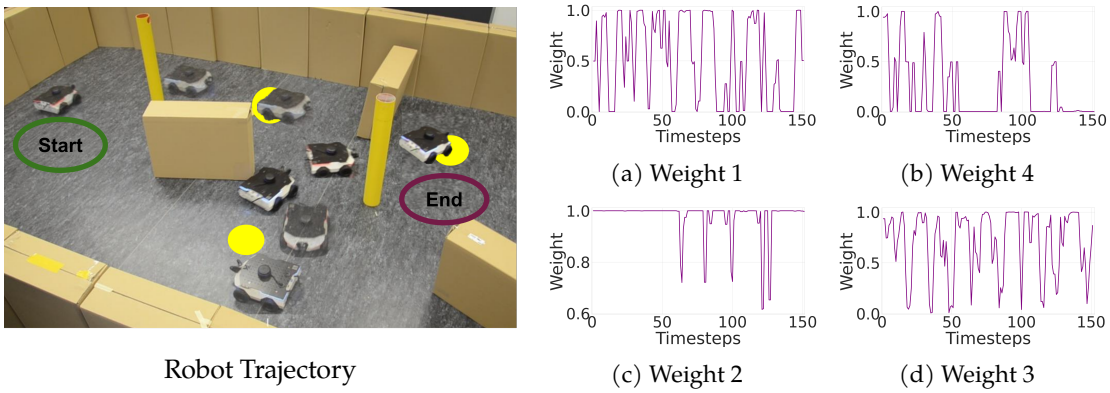
As mentioned previously, we do not provide any task information to the supervisor agent to minimize the exploration of the supervisor agent and hence minimize the number of collisions. In this section, we perform an ablation study in the first environment, where we add the goal information to the supervisor agent and give it a reward for reaching the goal and compare our approach without the goal information against the agent with the goal information. As can be seen from Fig. 5.4, introducing the goal information increases the number of collisions, as the added information expands the state space for the supervisor agent, resulting in a slower training process. This, in turn, reduces the goals-to-collisions ratio compared to our approach without the additional information. Nevertheless, the number of collisions remained lower than the constrained RL baselines.

5.5.3 Real-World Experiment

Finally, we investigate how the supervisor agent modifies the weights for the obstacles in a real-world experiment. Using checkpoints from our trained policies for both the task agent and the supervisor agent, we deployed these policies on a real robot. The policies were originally trained in simulation at 5 Hz, so we maintained the same control frequency on the robot to ensure consistency. We used the ROSbot XL [4] from Husarion for the experiments. The robot is equipped with a 360° lidar and an active tracker to track its location. See Fig. 5.5.

The supervisor agent, implemented as an RL policy, dynamically adjusts the weights of the obstacles based on the current environment and state information. These weights influence the importance assigned to specific obstacles when calculating safe actions. For instance, as the robot moves closer to an obstacle, the supervisor agent increases the corresponding weight, prioritizing obstacle avoidance in the MPC shield. Conversely, weights for farther or less relevant obstacles may decrease, allowing the robot to focus on reaching its goal efficiently.

[4] https://husarion.com/manuals/rosbot-xl/overview/



Robot Trajectory

(a) Weight 1

(b) Weight 4

(c) Weight 2

(d) Weight 3

Figure 5.5: The figure illustrates the real-robot trajectory (left) and the weights adjusted by the supervisor agent (right). The robot navigates from the start position to the goals (yellow circles) while avoiding obstacles. The weight plots are arranged such that each weight corresponds to its respective quadrant in the lidar data. The supervisor agent increases the weights of obstacles as the robot moves closer to them, prioritizing obstacle avoidance in the MPC shield.

Fig.5.5 illustrates the robot's trajectory as it navigates toward marked goals while avoiding obstacles. The plot shows the real-time changes in weights as modified by the supervisor agent. These changes reflect the RL agent's decision-making process to balance obstacle avoidance with the task agent's goal-directed actions.

5.6 Conclusion

This chapter presents a novel safety shield approach for reinforcement learning (RL) in navigation tasks, designed to effectively balance safety and exploration. By combining the robustness of model predictive control (MPC) safety shields with the long predictive capabilities of RL, we introduced a dynamic system where a supervisor agent adjusts the weights for the obstacle avoidance terms and for aligning the MPC actions with the task agent's actions. Our experiments demonstrate the superiority of the proposed approach across diverse environments. The results show that our method achieves the highest goals-to-collisions ratio, significantly minimizing collisions compared to constrained RL methods, i.e., [39], [120]. Unlike classical MPC-based shields [38], which often over-constrains the RL agents, our method promotes exploration without compromising safety, enabling the task agent to achieve higher rewards in challenging scenarios. Moreover, compared to unconstrained RL methods, which maximize goals at the cost of collisions, our approach strikes a balance by reducing collisions while maintaining competitive performance in goal-reaching metrics. The effectiveness of the dynamic adjustment mechanism was evident in its ability to adapt to varying environmental complexities, ensuring fewer constraints violations without hindering the task agent's progress. This balance emphasizes the potential of integrating optimization-based methods with RL to address real-world challenges in safe exploration. While this adaptive MPC framework provides a robust solution to the exploration-safety trade-off defined in RQ3, its reliance on an online optimization solver motivates the transition

to a more computationally efficient and model-free safety mechanism in the following chapter.

6 Constraint-Aware Reinforcement Learning via Adaptive Action Scaling

Abstract

Moving beyond optimization-based safety shields, this chapter addresses RQ4 by introducing a learned regulator that leverages predicted constraint violations to modulate actions, eliminating the need for an online solver while preserving exploratory performance. Safe reinforcement learning (RL) seeks to mitigate unsafe behaviors that arise from exploration during training by reducing constraint violations while maintaining task performance. Existing approaches typically rely on a single policy to jointly optimize reward and safety, which can cause instability due to conflicting objectives, or they use external safety filters that override actions and require prior system knowledge. In this chapter, we propose a modular cost-aware regulator that scales the agent's actions based on predicted constraint violations, preserving exploration through smooth action modulation rather than overriding the policy. The regulator is trained to minimize constraint violations while avoiding degenerate suppression of actions. Our approach integrates seamlessly with off-policy RL methods such as SAC and TD3, and achieves state-of-the-art return-to-cost ratios on Safety Gym locomotion tasks with sparse costs, reducing constraint violations by up to 126 times while increasing returns by over an order of magnitude compared to prior methods.

6.1 Introduction

Reinforcement Learning (RL) has demonstrated remarkable success across a range of domains, including Atari games [10], robotics [128], [129], and long-horizon strategy games [11], [12]. This success is significantly facilitated by exploratory behavior, which allows agents to discover effective behaviors. However, such exploratory behaviors often lead to the violation of constraints imposed on the controlled system. While constraint violations are tolerable in simulated environments and games where resets are free, they pose serious risks in real-world applications [130]. Violating safety constraints can lead to irreversible damage or system failure. To address this issue, Safe Reinforcement Learning (Safe RL) [16] has emerged as a critical area of research that aims to minimize constraint violations during both training and deployment.

Safe RL methods can be broadly categorized into two groups: *safe exploration* and *constrained RL*. Safe exploration techniques aim to prevent the agent from taking actions that violate safety constraints. These methods typically rely on prior knowledge of the system dynamics and feasible safe states to construct control barrier functions [105], [119], [131], or model predictive shields [38], [132], [133]. Although effective, their applicability is limited by the need for detailed prior information about the system dy-

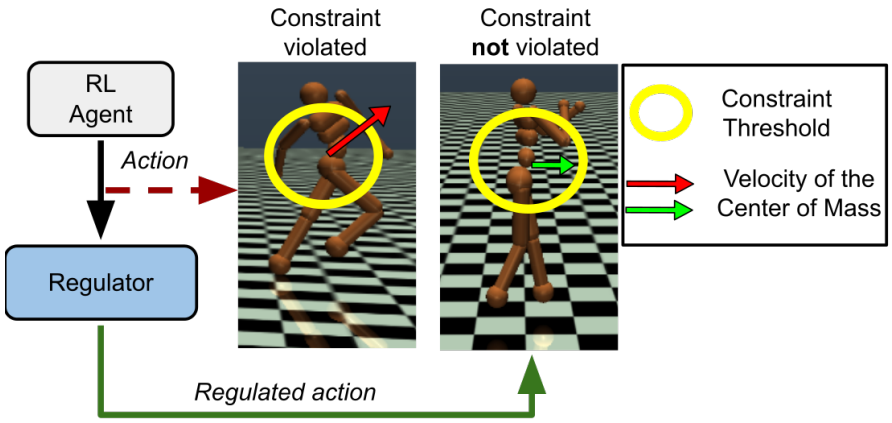


Figure 6.1: **Overview of cost-aware action scaling.** The RL agent proposes an action that would result in the center of mass (COM) exceeding the velocity threshold (left). The regulator (blue) intervenes by scaling the action, keeping the velocity of the COM within the safe zone while allowing progress on the task. The yellow circles highlight the velocity threshold for the COM, illustrating how the regulator enforces a safety constraint while preserving task performance.

namics, an assumption that often does not hold in early learning stages or tasks where system dynamics are unknown.

Constrained RL instead allows the agent to learn both reward and cost signals online, without requiring knowledge of the system dynamics. The agent is trained to maximize cumulative rewards while minimizing constraint violations. Common approaches include Lagrangian-based methods that solve the dual formulation of the constrained optimization problem [39], [40], [116], [120], and budget-based methods maintain a running estimate of remaining safety allowance, adjusting the agent's behavior to respect cost limits over time [122], [134].

However, a core limitation of these methods is the difficulty of balancing reward and cost within a single policy. Conflicting gradients can cause the agent to behave either too conservatively or unsafely, leading to instability, constraint violations, or poor performance [39], [135].

In contrast to prior work, we propose a modular alternative: instead of overriding actions or jointly optimizing conflicting objectives, we scale actions based on the expected cost of future constraint violations while preserving the policy's task-directed behavior (see Fig. 6.1).The architecture consists of a reward-maximizing task agent and a regulator network guided by twin cost critics to conservatively estimate constraint violations. The regulator applies element-wise scaling to attenuate risky actions, enforcing safety without requiring prior knowledge of dynamics or compromising exploration. Although our approach resembles safe exploration in formulation, we do not require prior knowledge of the system dynamics, and we do not override the task agent's actions, thereby preserving both exploration and safety without external overrides.

We evaluate our approach on several dynamical systems from Safety Gymnasium [136] and the Safety-Critical environments from [137]. Our method achieves the highest Return-to-Cost (RC) ratio [121], reducing constraint violations by up to

126 times over recent safe RL baselines [39], [134], [138], [139], [140]. In summary, our contributions are:

- We propose a modular safe RL framework that decouples reward maximization and safety enforcement via a cost-aware regulator that scales actions based on predicted violations.
- Our model-free approach integrates seamlessly into standard off-policy RL pipelines such as SAC [75] and TD3 [78], improving safety without compromising exploration.
- We achieve state-of-the-art performance on safety benchmarks, with up to 126 times fewer constraint violations and the highest RC ratios across tasks.

6.2 Related Work

Safe reinforcement learning (Safe RL) aims to enable agents to maximize task rewards while minimizing constraint violations. Existing approaches broadly fall into two categories: *safe exploration methods*, which intervene externally to prevent unsafe actions during training, and *constrained RL methods*, which embed cost objectives directly into policy optimization.

6.2.1 Safe Exploration Methods.

Safe exploration techniques intervene during training to prevent agents from entering unsafe states. Early methods, such as [141], employed uncertainty modeling through Gaussian Processes to restrict exploration and ensure safety during optimization. Later approaches introduced safety layers [91], [118] and predictive safety filters [38], [132], [133], [137] that anticipate and block risky actions based on pre-trained layers or model predictive control (MPC). Control Barrier Function (CBF)-based strategies [105], [119], [131] encode safety constraints directly through differentiable control barriers to guarantee that the agent's actions remain within certified safe sets throughout training. Recovery RL [121] assumes access to an offline dataset for pretraining a constraint critic, which is then fixed during online learning, limiting its applicability in settings where collecting sufficient offline data is challenging or costly. While our method shares some conceptual similarity with these safe exploration approaches—modifying actions to maintain safety—it differs fundamentally by relying on online-learned cost predictions rather than external models or handcrafted safe sets, and by smoothly scaling actions instead of hard blocking or overwriting them, preserving the agent's exploratory behavior.

6.2.2 Constrained RL Approaches.

Constrained RL methods integrate cost minimization into policy learning itself. Lagrangian-based algorithms [39], [40], [120] optimize dual formulation balancing

rewards and costs, while budgeted RL approaches [122], [134] include the remaining cost budget in the state representation, allowing the agent to adapt its behavior based on how much cost it can still afford. Risk-sensitive formulations, such as CVaR-CPO [142], enforce safety by constraining the conditional value-at-risk of cumulative costs, ensuring attention to costly violations. Reachability-based methods like RESPO [139] estimate the probability of reaching safe regions and optimize policies to persistently satisfy constraints or recover when outside the feasible set. Constraint-Conditioned Policy Optimization [143] enables zero-shot generalization to unseen cost thresholds by conditioning the policy and value functions on constraint levels using a variational inference objective. Bi-level optimization frameworks such as SRCPO [140] address the nonlinearity of risk measures by optimizing over dual variables, achieving strong constraint satisfaction in continuous control tasks. Compared to these methods, our approach offers a lightweight, modular alternative: instead of embedding constraints into the policy loss or relying on delicate dual updates, we regulate actions externally based on learned cost estimators, allowing standard reward-driven optimization to proceed unmodified.

The most closely related work to ours is Safety Editor [138], which trains two separate Soft Actor-Critic (SAC) [75] agents: a utility maximizer and a safety editor that modifies unsafe actions. The safety editor is trained to minimize a hinge loss that penalizes reductions in utility Q-values while satisfying learned safety constraints. As a result, it can fully overwrite the original action when necessary. In contrast, our approach uses a lightweight multilayer perceptron (MLP) to predict a scaling factor that continuously modulates the magnitude of the original action based on predicted cost estimates. This design maintains the agent's ability to explore effectively while attenuating risky actions, enabling modular integration into standard off-policy RL pipelines without the need for a separate actor.

6.3 Preliminaries

6.3.1 Markov Decision Processes.

We consider a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(s'|s,a)$ the transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in (0,1)$ the discount factor. We assume continuous state and action spaces with $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^d$.

6.3.2 Constrained Reinforcement Learning.

Constrained RL refers to the problem of maximizing task rewards while satisfying explicit constraints on agent behavior. The standard formalism is through Constrained Markov Decision Processes (CMDPs) [144], where constraints are expressed using cost functions separate from the reward.

6.3.3 Constrained Markov Decision Processes.

A CMDP augments the MDP with a cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ that quantifies safety violations. The objective is to maximize return while keeping the expected cumulative cost below a budget χ:

$$\max_{\pi} \ \mathbb{E}_{s\sim d_0,\, a\sim\pi(\cdot|s)}[Q^{\pi}(s,a)] \quad \text{s.t.} \quad \mathbb{E}_{s\sim d_0,\, a\sim\pi(\cdot|s)}[Q_c^{\pi}(s,a)] \leq \chi. \tag{CMDP}$$

where the cost value functions are

$$V_c^{\pi}(s) = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \,\Big|\, s_0 = s\Big], \tag{6.1}$$

$$Q_c^{\pi}(s,a) = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \,\Big|\, s_0 = s,\, a_0 = a\Big]. \tag{6.2}$$

6.3.4 Cost Budget.

The budget χ specifies the maximum allowable expected cumulative cost and is typically treated as a human-selected threshold that reflects task-specific safety requirements [39]. In this work, we assume a stricter setting by eliminating the cost budget, i.e., setting $\chi = 0$, similar to [139]. This corresponds to a hard-safety regime that aims to achieve minimal constraint violations during learning.

6.3.5 Problem Setting.

With this stricter formulation, the problem considered in this work is to learn a policy that maximizes task rewards while minimizing constraint violations under the hard-safety regime $\chi = 0$. Formally, our objective reduces to:

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}_{s\sim d_0,\, a\sim\pi(\cdot|s)}[Q^{\pi}(s,a)] \quad \text{s.t.} \quad \mathbb{E}_{s\sim d_0,\, a\sim\pi(\cdot|s)}[Q_c^{\pi}(s,a)] = 0. \tag{6.3}$$

This assumption eliminates any positive cost budget and focuses on policies that aim to achieve minimal safety violations during training and execution.

6.4 Our Approach

We propose a modular safe reinforcement learning framework that regulates the actions of a task policy to reduce expected constraint violations without overriding agent decisions. The key idea is to scale actions based on their predicted cost, preserving exploration while inducing smoother and safer transitions in the environment.

6.4.1 Split Architecture for Reward and Cost Optimization

Optimizing for both task rewards and safety constraints within a single policy often leads to instability or overly conservative behavior [39]. To address this, we decouple the

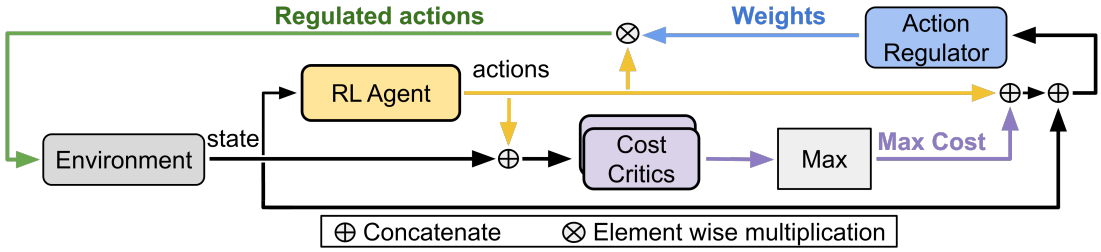


Figure 6.2: **Overview of our modular safe RL architecture**. The regulator (blue) scales actions produced by the unconstrained RL agent (yellow) based on predicted cost (purple), producing safety-aware actions (green) that are executed in the environment.

reward and cost learning objectives across two modules: The **task policy** $\pi_\phi(a|s)$, which is trained to maximize expected rewards without incorporating safety constraints. The **regulator network** $\rho_\theta(s, a, \hat{c})$, which learns to scale the policy's actions based on cost predictions to minimize constraint violations.

6.4.2 Action Modulation via Regulator Scaling

In continuous control environments without stochasticity, the system evolves under deterministic transition dynamics of the form $s_{t+1} = f(s_t, a_t)$, where $s_t \in \mathcal{S}$ is the current state and $a_t \in \mathcal{A} \subset \mathbb{R}^d$ is a d-dimensional real-valued action vector, with d denoting the number of action dimensions. Since actions directly control the system evolution, high-magnitude or poorly directed actions can result in constraint violations or unstable behaviors. To mitigate this, we introduce a *regulator network* $\rho_\theta : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to (0, 1]^d$, which learns a scaling vector with an individual factor for each action dimension based on the current state, the raw action, and its predicted cost. At each step, the agent samples a raw action $a_t \sim \pi_\phi(\cdot|s_t)$, computes the cost estimate: $\hat{c}_t = \max(Q_c^1(s_t, a_t), Q_c^2(s_t, a_t))$ from a twin-critic architecture, and the regulator network outputs a scaling vector ρ_t, see Fig. 6.2. The final action applied to the system is:

$$\tilde{a}_t = \rho_t \odot a_t, \quad \text{where } \rho_t = \rho_\theta(s_t, a_t, \hat{c}_t), \tag{6.4}$$

where $\odot$ denotes element-wise multiplication, where each component of the action vector is multiplied by a scaling factor between 0 (high risk, large attenuation) and 1 (safe, no attenuation), smoothly reducing potentially unsafe actions proportional to predicted risk. This element-wise modulation attenuates each component of the action based on its risk profile, reducing the magnitude of high-risk components. Unlike hard safety constraints that may override agent behavior, this approach preserves the agent's exploration behavior and allows stable off-policy learning. In many robotic domains, safety costs naturally increase with the magnitude of control inputs: high torques in manipulators accelerate wear and overheating, and large contact forces in legged robots risk joint damage. By designing costs that capture this structure, practitioners can align the regulator with system-specific safety considerations, making scaling an intuitive and

broadly applicable mechanism for enforcing safety.

6.4.3 Learning Objectives and Updates

Reward Learning. We adopt a general off-policy reinforcement learning framework where the agent's actor and critic are trained using the scaled action $\tilde{a}_t$, as this is the action that is actually executed in the environment. The reward critic is updated using:

$$Q_r(s_t, \tilde{a}_t) \leftarrow r(s_t, \tilde{a}_t) + \gamma \, \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, \tilde{a}_t), a_{t+1} \sim \pi(\cdot|s_{t+1})} \left[Q_r(s_{t+1}, \tilde{a}_{t+1}) \right], \tag{6.5}$$

where $\tilde{a}_{t+1} = \rho_{t+1} \odot a_{t+1}$ and $\rho_{t+1} = \rho_\theta(s_{t+1}, a_{t+1}, Q_c(s_{t+1}, a_{t+1}))$. The policy is updated to maximize the expected return under the regulated action:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi(\cdot|s_t)} \left[-Q_r(s_t, \tilde{a}_t) \right], \tag{6.6}$$

ensuring that policy learning reflects the actual dynamics induced by the regulated action $\tilde{a}_t$. Our framework is algorithm-agnostic and can be integrated with any off-policy actor-critic method. For entropy-regularized algorithms such as SAC, the corresponding entropy term may be included in the actor objective. In our experiments, we demonstrate compatibility with both SAC and Twin Delayed DDPG (TD3)[78].

Cost Learning. The cost critic is also trained on the scaled actions using a TD-style Bellman backup [9]:

$$Q_c(s_t, \tilde{a}_t) \leftarrow c(s_t, \tilde{a}_t) + \gamma \, \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, \tilde{a}_t), a_{t+1} \sim \pi(\cdot|s_{t+1})} \left[Q_c(s_{t+1}, \tilde{a}_{t+1}) \right]. \tag{6.7}$$

This ensures the critic reflects the safety implications of the actual executed action $\tilde{a}_t$.

Regulator Objective. The regulator is trained to minimize the predicted cost of the executed action $\tilde{a}_t$, while avoiding degenerate solutions that collapse actions toward zero. Its loss function is given by:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi(\cdot|s_t)} \left[\beta \cdot Q_c(s_t, \tilde{a}_t) - \lambda \cdot \log \rho_\theta(s_t, a_t, \hat{c}_t) \right], \tag{6.8}$$

where $\beta, \lambda > 0$ are trade-off parameters. The first term encourages the regulator to scale down actions that lead to high predicted costs. However, without the second term, a trivial solution where $\rho_\theta(s, a, \hat{c}) \to 0$ would minimize this objective by collapsing all actions—halting the agent's behavior entirely. To counteract this, the second term acts as a *barrier penalty* that diverges as any element of the scaling vector approaches zero. It encourages the regulator to retain as much of the original action magnitude as possible, unless high predicted cost necessitates suppression.

Optimality Trade-Off. The regulator's training objective can be interpreted as solving a local constrained optimization problem at each state-action pair (s_t, a_t):

$$\min_{\rho \in (0,1]^d} \; \beta \cdot Q_c(s_t, \rho \odot a_t) - \lambda \cdot \log(\rho + \epsilon), \tag{6.9}$$

where the logarithm is applied element-wise to the scaling vector ρ, and we include ϵ to avoid instability as $\rho \to 0$, ensuring gradients remain well-defined during training.

The coefficients β and λ balance the trade-off between minimizing predicted cost and preserving action magnitude: larger λ encourages less suppression, while larger β prioritizes cost reduction. Since $Q_c(s, \rho \odot a_t)$ is typically a nonlinear function of the scaled action, the optimization problem lacks a closed-form solution but can be efficiently solved via gradient-based updates. This formulation ensures that the regulator selectively attenuates risky action dimensions while retaining as much of the agent's original behavior as possible.

Gradient Flow and Modularity. To ensure clean modularity, we detach the scaling weights $\rho_\theta(s_t, a_t, \hat{c}_t)$ from the computational graph when updating both the reward and cost critics, preventing gradients from flowing through the regulator. Similarly, the actor receives no gradients from the regulator, learning purely from task returns. Moreover, the regulator is updated independently via its own objective, ensuring that reward maximization and safety modulation remain decoupled. The full training procedure is summarized in Algorithm 1.

Algorithm 1 Cost-Aware Action Scaling (training loop)

Require: Environment $\mathcal{E}$, replay buffer $\mathcal{D}$, actor π_ϕ, regulator ρ_θ, reward critic Q_r, cost critic Q_c

Initialize network parameters and target networks
for each interaction step **do**
  Observe s and sample $a \sim \pi_\phi(\cdot \mid s)$
  $\hat{c} \leftarrow \max\big(Q_c^1(s,a),\, Q_c^2(s,a)\big)$ ▷ predicted cost
  $\rho \leftarrow \rho_\theta(s, a, \hat{c})$; $\quad \tilde{a} \leftarrow \rho \odot a$ ▷ scaled action
  Execute $\tilde{a}$, obtain (r, c, s'), store $(s, \tilde{a}, r, c, s')$ in $\mathcal{D}$
for each gradient step **do**
  Sample minibatch from $\mathcal{D}$
  Update Q_r and π_ϕ via Eqs. (6.5)–(6.6)
  Update Q_c (Eq. 6.7)
  Update ρ_θ (Eq. 6.8)
  Polyak-average target networks

This design is particularly well-suited for **off-policy reinforcement learning**, where updates are performed using transitions stored in a replay buffer, independent of the current policy. Since the regulator modulates actions *after* sampling from the policy $\pi_\phi(\cdot \mid s)$, the executed action $\tilde{a} = \rho_\theta(s, a, \hat{c}) \odot a$ differs from the originally sampled action a, and only the regulated action is stored and used for training. Off-policy methods naturally accommodate this, as policy and critic updates rely on the actual executed actions rather than the distribution used to generate them. By contrast, **on-policy methods** such as trust region policy optimization (TRPO) [145] and proximal policy optimization (PPO) [146] require that training actions be drawn directly from the current policy. Post-sampling modifications like our regulator violate this assumption, disrupting likelihood ratio estimation and invalidating surrogate objectives. Supporting such integration would require embedding the regulator within the sampling process, thereby breaking the clean modular separation we aim to preserve. Hence, our approach is in-

herently better suited to off-policy learning.

Implementation Details. The proposed method integrates a standard off-policy RL agent with a lightweight action regulator network for constraint satisfaction. The RL agent follows its baseline implementation without modification. The regulator is a feedforward neural network with two hidden layers of sizes [256, 256] and ReLU activations, outputting element-wise scaling factors between 0 and 1 via a sigmoid activation to modulate the agent's actions. The regulator is trained using predicted costs from twin cost critics. Each cost critic is a feedforward network with two hidden layers of sizes [256, 256] and Tanh activations, taking state-action pairs as input and outputting a scalar cost prediction.

6.5 Experiments

Our experiments are designed to achieve the following objectives: (i) compare our approach against state-of-the-art safe RL baselines across different dynamical systems, (ii) analyze the influence of the key hyperparameters (λ and β) from Eq. 6.9, which govern the trade-off between action preservation and cost suppression, (iii) evaluate the regulator's action-scaling mechanism through ablations such as element-wise versus scalar regulation on different systems, (iv) investigate the behavior of the scaling mechanism on the actuation in one of the systems, such as the Ant robot, and (v) study robustness under injected sensor and actuator noise, highlighting the method's potential for sim-to-real transfer.

6.5.1 Environments

We evaluate our method on locomotion tasks from the **Safety Gym** benchmark [136], namely `Ant`, `Walker2d`, `Swimmer`, `HalfCheetah`, and `Humanoid`. In these velocity tasks, a safety cost is incurred whenever the center-of-mass speed exceeds a predefined threshold. Because the cost signal is sparse and triggered only by such threshold violations, these environments provide a challenging setting for safe RL, while naturally aligning with our regulator's goal of attenuating large actions that are most likely to induce violations. We further evaluate on `BiGlucose` and the `Continuous Stirred Tank Reactor (CSTR)` from the **Safety-Critical Systems** [137].

6.5.2 Baselines

We compare our regulator against five state-of-the-art Safe RL baselines. **PPO-PID** [39] augments PPO with a PID-controlled Lagrangian multiplier to mitigate instabilities commonly observed in dual updates during constrained optimization. **Simmer** [134] augments PPO with a safety state that tracks the remaining safety budget. **Safety Editor** [138] uses two SAC agents, one for maximizing the reward and another for editing unsafe actions. **RESPO** [139] estimates reachability sets and constrains policy to remain within safe regions. **SRCPO** [140] formulates a bi-level constrained optimization using

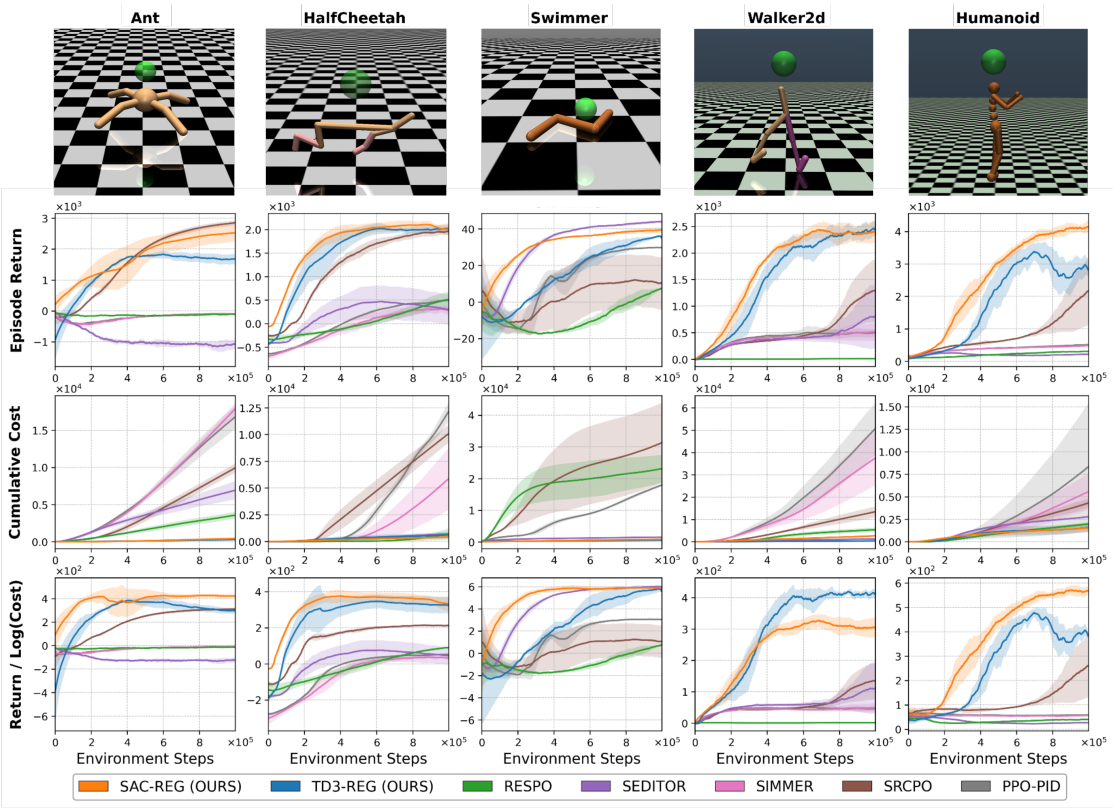


Figure 6.3: Performance comparison on the Safety Gymnasium locomotion environments. Each method is averaged over three independent runs; bold lines indicate the mean, and shaded areas show the standard deviation. Our methods (SAC-REG and TD3-REG) consistently achieve the best trade-off between return and cumulative constraint cost across all environments. **Top:** Episode return. **Middle:** Cumulative safety cost. **Bottom:** Return-to-Log-Cost ratio. Our methods outperform strong baselines, including PPO-PID [39], SIMMER [134], SRCPO [140], RESPO [139], and SEDITOR [138]. SIMMER is omitted from the `Swimmer` plot as it consistently yields negative returns, moving opposite to the target velocity.

spectral risk measures to achieve a near-zero constraint violation rate while maximizing reward.

6.5.3 Metrics

Similar to prior safe RL studies, we report returns and cumulative costs as in [121], [139], [147], and additionally follow [121] in using the return-to-cost (RC) ratio to capture the trade-off between task performance and safety. Specifically, we measure (i) episodic return, (ii) cumulative cost during training, which reflects the total number of constraint violations and, in sparse-cost settings such as Safety Gym velocity tasks, implicitly captures violation frequency, and (iii) the RC ratio, defined as the total return divided by cumulative cost. For visualization, we plot the return divided by the logarithm of the accumulative cost, which improves interpretability of the safety-performance trade-off.

Relative Return Improvement of SAC-Reg Over Baselines (↑)					
Method	Ant	HalfCheetah	Swimmer	Walker2d	Humanoid
PPO-PID [39]	27.33	3.04	0.32	3.76	7.13
SIMMER [134]	26.14	6.09	1.21	3.59	7.60
SEditor [138]	3.34	5.64	-0.10	1.95	18.52
RESPO [139]	23.97	2.89	0.17	204.40	12.73
SRCPO [140]	-0.11	0.02	2.90	0.86	0.90
Relative Cost Compared to SAC-Reg (↓)					
PPO-PID [39]	39.29	28.39	21.31	19.11	5.46
SIMMER [134]	41.88	13.70	54.22	14.14	3.64
SEditor [138]	16.19	1.67	1.77	0.48	1.88
RESPO [139]	8.36	1.65	27.39	2.03	1.29
SRCPO [140]	23.24	23.55	37.10	5.07	2.82

Table 6.1: Relative return improvement (↑) and relative cumulative cost (↓) of SAC-Reg compared to baselines across locomotion tasks. SAC-Reg consistently achieves higher returns and lower cumulative costs than prior safe RL methods across all environments.

Relative Return Improvement of TD3-Reg Over Baselines (↑)					
Method	SafetyAnt	HalfCheetah	Swimmer	Walker2d	Humanoid
PPO-PID [39]	18.49	3.05	0.17	3.87	4.51
SIMMER [134]	17.70	6.11	1.18	3.70	4.83
SEditor [138]	2.55	5.65	-0.20	2.02	12.22
RESPO [139]	16.26	2.90	3.59	209.23	8.30
SRCPO [140]	-0.41	0.02	2.46	0.91	0.29
Relative Cost Compared to TD3-Reg (↓)					
PPO-PID [39]	60.14	19.77	34.06	126.18	5.20
SIMMER [134]	64.11	9.54	86.65	93.37	3.47
SEditor [138]	24.78	1.16	2.84	3.15	1.74
RESPO [139]	12.80	1.15	43.78	13.42	1.23
SRCPO [140]	35.57	16.39	59.28	33.50	2.69

Table 6.2: Relative return improvement (↑) and relative cumulative cost (↓) of TD3-Reg compared to baselines across locomotion tasks.TD3-Reg demonstrates similar trends, outperforming baselines in return while maintaining substantially lower cumulative costs.

6.5.4 Comparison Against Baselines:

6.5.4.1 Safety Gym Results

Across the locomotion tasks in the Safety Gymnasium suite, our methods—**SAC-Regulator** and **TD3-Regulator**—consistently deliver strong task performance while substantially reducing safety violations. Each method was evaluated over three random seeds; bold lines in Fig.6.3 denote the mean return across runs, with shaded regions representing standard deviation. Compared to the baselines, our approach achieves higher

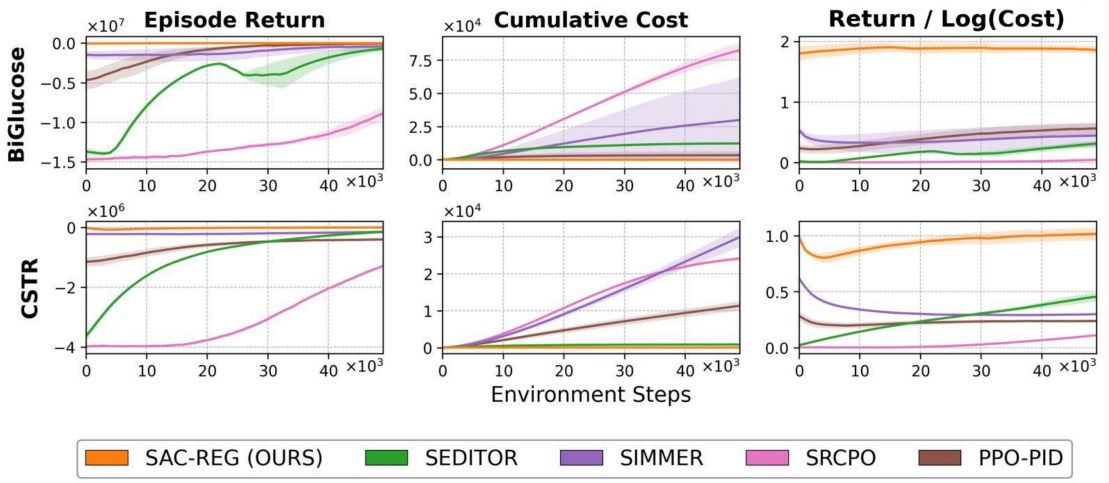


Figure 6.4: Performance comparison on the BiGlucose and CSTR environments. Our method (SAC-Reg) achieves high return with low cost, yielding the best return-to-cost ratio throughout training.

or comparable episode returns, indicating that soft action scaling does not hinder exploration. Notably, **TD3-Regulator** achieves the greatest cost reductions, with up to **126×** lower cumulative cost in `Walker2d`, **64×** in `Ant`, and **86×** in `Swimmer`. Meanwhile, **SAC-Regulator** outperforms both **TD3-Regulator** and all baselines in `HalfCheetah` and `Humanoid`, achieving cost reductions of up to **28×** and **5×**, respectively. RESPO can reach comparable returns when trained for 9M steps, but only with substantially higher violations and failure to converge in `Humanoid`.

Tables 6.1 and 6.2 summarize results for both regulators against established baselines. Two metrics are reported: the *relative return improvement*,

$$\frac{\text{Return}_{\text{Ours}} - \text{Return}_{\text{Baseline}}}{|\text{Return}_{\text{Baseline}}|},$$

and the *relative cumulative cost* of each baseline normalized by our method. Positive return values indicate improved task performance, while cumulative cost ratios above 1.0 indicate higher constraint violations than ours. For example, in `Walker2d`, **SAC-Reg** outperforms RESPO with a relative return improvement of 204.4, corresponding to a 20,440% increase.

Overall, our methods deliver the lowest cumulative cost across all tasks without compromising return. Unlike approaches such as SEditor or RESPO, which improve safety at the expense of performance, our regulators preserve exploration and consistently achieve superior return-to-cost trade-offs. This demonstrates the effectiveness and generality of decoupling safety regulation from reward learning.

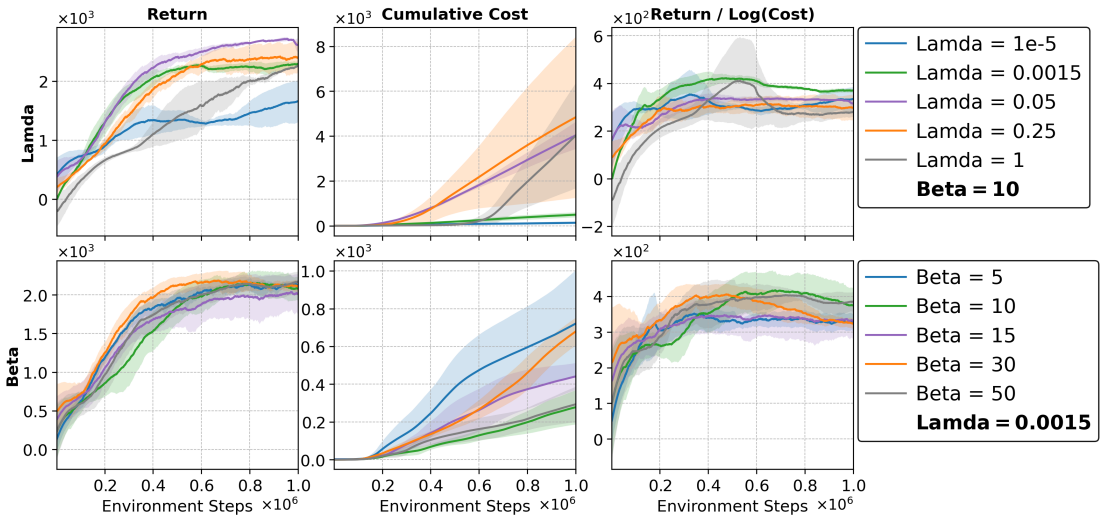


Figure 6.5: Ablation Study for evaluating the impact of the regulator hyperparameters λ and β on Return, Cumulative Cost, and Return-to-Cost ratio. Each curve shows the mean across three runs, and shaded regions indicate standard deviation. The top row varies λ with fixed $\beta = 10$; the bottom row varies β with fixed $\lambda = 0.0015$. As seen, smaller λ values reduce cumulative cost, with $\lambda = 0.0015$ giving the best balance between performance and safety, while $\beta = 10$ provides the most favorable trade-off overall.

6.5.4.2 Experiments on Safety Critical Systems

We evaluate our approach on two continuous-control environments: `BiGlucose` and the `Continuous Stirred Tank Reactor (CSTR)` from [137], which capture biomedical and chemical process dynamics. `BiGlucose` models blood glucose regulation with insulin and glucagon injections under delayed, partially observable dynamics, requiring glucose levels to stay within physiological bounds. `CSTR` simulates nonlinear reactor dynamics where unsafe control can cause hazardous runaway reactions. In both cases, we modify the environments to provide a continuous cost signal by logging violation magnitudes, instead of terminating episodes on constraint breaches. For full specifications, see [137].

Figure 6.4 compares our **SAC-Reg** method against baselines (SEditor, SIMMER, SRCPO, and PPO-PID). All methods are trained for the same number of environment steps as in [137]; RESPO is excluded since it takes significantly more steps to converge. Across both tasks, **SAC-Reg** consistently achieves higher returns with fewer cumulative constraint violations, yielding superior safety–performance trade-offs. In BiGlucose, baselines such as SIMMER and SRCPO quickly accumulate costs despite improving return, while SEditor remains return-limited. In CSTR, only our method manages both safety and performance, as others either accumulate high violations or fail to learn.

6.5.5 Ablation Study on λ and β

We conduct an ablation study in the `Ant` environment to evaluate the sensitivity of our regulator framework to the hyperparameters λ and β, which control the trade-

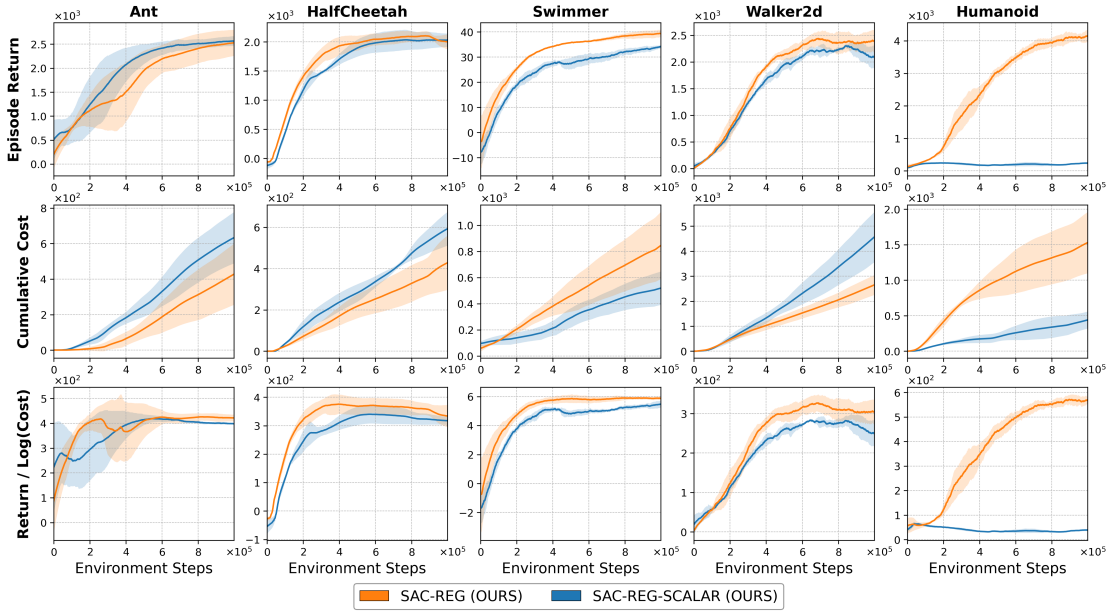


Figure 6.6: Comparison between element-wise and scalar action scaling. While both perform similarly in most tasks, the scalar variant fails to converge in the high-dimensional `Humanoid` environment, indicating that element-wise scaling improves safety and stability in complex control settings.

off between action retention and cost suppression. When varying λ over the range $\{1 \times 10^{-5},\ 0.0015,\ 0.05,\ 0.25,\ 1.0\}$, we find that smaller values lead to significantly lower cumulative costs. In particular, $\lambda = 0.0015$ achieves the best balance between constraint satisfaction and task performance. Higher values of λ result in larger action magnitudes and consequently higher constraint violations. Similarly, varying β over $\{5,\ 10,\ 15,\ 30,\ 50\}$ shows that $\beta = 10$ achieves the best overall safety-performance balance, minimizing constraint violations while maintaining high return. These results, as shown in Fig. 6.5, highlight the importance of properly tuning the regulator's loss coefficients to achieve optimal return-to-cost behavior.

6.5.6 Element-wise vs. Scalar Regulation

To evaluate the impact of element-wise action regulation, we conducted an ablation study comparing our full regulator with a simplified variant that uses a single scalar value to uniformly scale all action dimensions. Figure 6.6 presents results across all Safety Gymnasium locomotion tasks. While the scalar variant achieves comparable performance in most environments, it fails to converge in the high-dimensional `Humanoid` task. This suggests that element-wise scaling is particularly important in complex control settings, where individual action dimensions exhibit distinct risk profiles. Fine-grained modulation allows the regulator to target risky joints more precisely, improving both safety and learning stability.

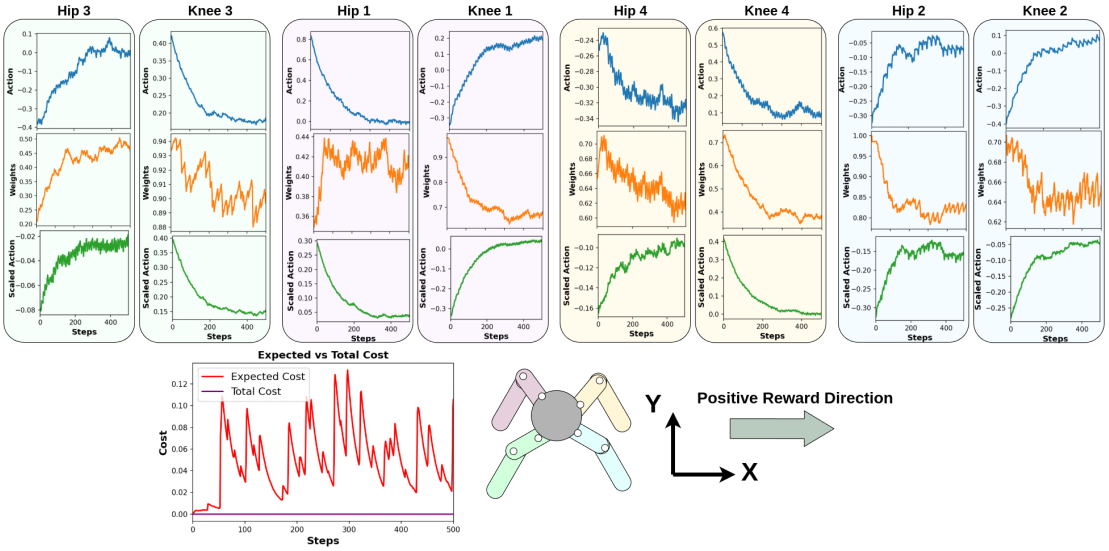


Figure 6.7: **Actuation patterns after training SAC with our regulator (500 steps).** Each column corresponds to a joint on the SafetyAnt environment: showing raw action (top), learned regulator weight (middle), and scaled action (bottom). The regulator selectively attenuates high-risk action components based on predicted safety cost. The bottom-left plot compares expected (red) vs. actual (purple) cost across steps: illustrating early conservative estimation by the regulator. The ant figure illustrates the robot's limb-to-color mapping and the direction of positive task reward.

6.5.7 Regulator Behavior Analysis.

To investigate the behavior of the scaling mechanism in detail, we visualize the regulator's modulation patterns on the `SafetyAntVelocity-v1` environment after training with SAC-Regulator. Figure 6.7 shows the raw actions, scaling weights, and resulting scaled actions for each joint over 500 steps. This per-joint illustration highlights how the regulator dynamically suppresses high-risk components when expected costs are high. Importantly, the regulator does not uniformly scale down all actions. Instead, it selectively attenuates only the high-risk components—those expected to incur higher constraint costs—while leaving low-risk components largely unchanged. This allows the robot to continue progressing toward its task objective while minimizing constraint violations. It also demonstrates that the regulator adapts effectively to local cost signals while maintaining coordinated control across joints.

6.5.8 Robustness and Sim-to-Real Transfer

To approximate uncertainties encountered on physical robots, we inject Gaussian noise into both observations and actions during training, modeling sensor measurement errors and actuator execution noise. Agents are trained with noise levels ($\sigma = 0, 0.025, 0.05, 0.10$), and the resulting training performance is summarized in Table 6.3. Across all noise settings, our regulator achieves strong returns while keeping cumulative costs bounded. Even under the highest noise level ($\sigma = 0.10$), performance remains stable, highlighting robustness to sensing and actuation imperfections and supporting

Step	Noise 0.00		Noise 0.025		Noise 0.05		Noise 0.10	
	Return	Cost	Return	Cost	Return	Cost	Return	Cost
100k	1961	6	753	0	762	28	677	1
300k	2558	79	1554	78	1614	165	1103	113
500k	2492	199	1945	195	1969	282	1381	387
700k	2533	282	2139	230	2071	339	1596	572
900k	2501	388	2104	262	2074	479	1589	767
1000k	2571	412	2016	312	2089	533	1510	849

Table 6.3: Training performance under different levels of injected Gaussian noise ($\sigma = 0.00, 0.025, 0.05, 0.10$) in observations and actions. Values show episode return and cumulative cost at checkpoints. Our regulator maintains bounded costs across noise levels, demonstrating robustness relevant for sim-to-real transfer.

the method's potential for sim-to-real transfer.

6.6 Conclusion

We introduced a modular and practical framework for safe reinforcement learning that decouples reward maximization from safety enforcement through a cost-aware regulator. Instead of overriding agent actions, our method scales them smoothly based on predicted constraint violations, preserving exploration and enabling stable off-policy learning. The regulator uses twin cost critics for robust cost estimation and is trained with a loss that balances risk reduction and action preservation. Our approach is model-free and integrates seamlessly with existing off-policy RL pipelines. Empirical results on diverse benchmarks demonstrate that our method consistently achieves the highest return-to-cost ratios, reducing constraint violations by up to 126 times while increasing returns by over an order of magnitude compared to prior state-of-the-art methods. The regulator aligns with real-world safety limits such as torque bounds in manipulators, and joint load management in legged robots. Robustness experiments with injected observation and action noise further demonstrate bounded costs and stable returns under uncertainty, supporting the potential for sim-to-real transfer. A key direction for future work is to develop principled strategies for automatically tuning the regulator hyperparameters (λ and β) and to extend the approach beyond input-magnitude costs toward more general safety constraints.

7 Conclusion and Outlook

7.1 Conclusion

In this thesis, we included prediction as a framework for exploration: we combined reinforcement learning with model predictive control and learned safety signals to make training safer and more sample-efficient. Across four interconnected studies, we moved from (i) using MPC as an experience source under sparse rewards, to (ii) enforcing safety in cooperative multi-robot navigation with distributed NMPC filters, (iii) adapting a safety shield online via a supervisor agent, and (iv) designing a modular regulator that scales actions based on predicted constraint violations. Together, these results show that predictive structure can guide data collection, constrain risk without freezing exploration, and yield policies that successfully transfer from simulation to the physical world.

Accelerating Exploration with MPC Guidance (Chapter 3). Addressing the fundamental challenge of learning under sparse rewards, we proposed a hybrid training framework that utilizes MPC as a synthetic demonstrator. Unlike standard approaches that rely on costly human data or inefficient random exploration, our method interleaves MPC-generated trajectories with the agent's trial-and-error experience. We introduced a policy selector with a decaying schedule that leverages the controller's foresight during early training phases and progressively hands over control to the RL agent. Empirical results in static and dynamic navigation environments showed that this guidance significantly accelerated convergence and allowed the agent to discover strategies for complex obstacle avoidance that pure RL failed to find. Crucially, we demonstrated that the agent learns to navigate reliably at inference time without requiring the MPC solver.

Scalable Safety for Multi-Robot Teams (Chapter 4). Building on single-agent efficiency, we extended our focus to the safety and scalability of multi-agent systems. We introduced a novel formulation for behavior-based cooperative navigation that eliminates the need for individual reference targets, driving the team via a single formation centroid. To secure this high-interaction setting, we integrated distributed NMPC safety filters with an attention-based MARL architecture. This combination ensured zero collisions during both training and execution—a critical requirement for real-world deployment. A key finding of this study was the scalability of the approach: policies trained with only three agents successfully generalized to teams of eight robots without retraining. Furthermore, we validated the zero-shot transfer of these policies to physical robots, proving that the safety layer effectively bridges the sim-to-real gap even during the early stages of learning.

Adaptive Shielding for Robust Exploration (Chapter 5). While hard safety filters prevent collisions, they often over-constrain the exploration. To resolve this, we developed a dynamic safety shield that adapts constraint parameters online. We intro-

duced a hierarchical supervisor agent trained to tune the weights of obstacle avoidance and action-alignment terms in real-time based on environmental complexity. This approach addressed the limitations of static shields, which often suffer from solver failures in dense environments. Our results demonstrated that this adaptive mechanism achieves a superior trade-off between safety and task progress, yielding the highest goals-to-collisions ratio compared to constrained RL and static MPC baselines. Notably, we showed that the supervisor operates effectively without goal information, focusing purely on immediate safety and feasibility.

Efficient Safety via Constraint-Aware Regulation (Chapter 6). Finally, to eliminate the computational bottleneck of running optimization solvers at runtime, we proposed a modular Action Regulator. This method encapsulates the principles of predictive shielding into a lightweight neural network that preemptively scales actions based on risk estimates from twin cost critics. Unlike previous methods that override the policy or struggle with conflicting objectives, our regulator continuously modulates action magnitudes to satisfy constraints while preserving the policy's directional intent. Empirically, the method achieved state-of-the-art performance on safety benchmarks, reducing constraint violations by up to 126 times compared to baselines while operating with the inference speed of a standard neural network.

7.2 Limitations

While the proposed frameworks significantly advance the state of safe and efficient robot learning, several limitations remain inherent to the methods and assumptions adopted in this thesis. Acknowledging these constraints is essential for identifying directions for future research.

Reliance on Explicit Dynamics Models (Chapters 3–5). The MPC-based methods presented in Chapters 3, 4, and 5 rely on the availability of an explicit prediction model of the robot's dynamics. In our experiments, we utilized kinematic models which, while effective for ground mobile robots at moderate speeds, may not suffice for highly agile platforms or complex environments with significant slip, friction, or aerodynamic effects. If the internal model diverges significantly from the real-world dynamics (sim-to-real gap), the safety guarantees of the MPC shield diminish, potentially leading to unsafe behaviors before the RL agent can adapt. Extending these methods to model-free settings, incorporating online system identification, or adapting learning-based dynamic models remains an open challenge.

Conservative Behavior and Deadlocks (Chapter 4). In the distributed multi-robot setting, ensuring safety via decentralized filters can lead to deadlocks in dense scenarios. As observed in our generalization experiments, increasing the number of agents led to a higher rate of timeouts. This occurs because the safety filter operates based on instantaneous observations without knowledge of the future trajectories of neighbors, effectively treating them as static obstacles. Consequently, the filter may force agents

into conservative configurations where no feasible safe trajectory exists for any agent to proceed. While the dynamic shield (Chapter 5) mitigates over-conservatism for single agents, resolving distributed deadlocks in crowded multi-agent systems remains a complex coordination problem.

Assumptions of Action Regulation (Chapter 6). The learned Action Regulator proposed in Chapter 6 eliminates the online solver but introduces structural assumptions. The method relies on the assumption that reducing the magnitude of an action decreases safety risks. While valid for many robotic tasks, this assumption does not hold for systems where safety requires high-energy maneuvers, such as a drone needing thrust to maintain altitude or a legged robot needing velocity to clear a gap. Furthermore, unlike the hard constraints of MPC, the regulator provides soft, probabilistic safety; it requires encountering some constraint violations during training to learn the cost signal, making it less suitable for systems where even a single training violation is catastrophic.

7.3 Outlook

To conclude, we outline several extensions and open questions that follow directly from our results. These directions address the limitations identified in Sec. 7.2 by leveraging emerging techniques in latent dynamics, sequence modeling, and massively parallel simulation.

Learning Latent Dynamics for Convex Model-Free Safety (Extending Chaps. 3–5). A primary limitation of our current MPC shields is their reliance on explicit, hand-derived kinematic models. This restricts their applicability to systems with complex, highly nonlinear dynamics such as soft robots and legged robots. A promising direction is to generalize the safety shield using Koopman operator theory [148]. Koopman theory posits that nonlinear dynamics in a finite-dimensional state space can be lifted to an (infinite-dimensional) space in which their evolution is linear. By training a transformer-based architecture to learn such a linear embedding from data [149], we could approximate the global system dynamics as a linear system in a latent space. The key advantage is *convexity*: with a linear latent model and convex costs and constraints, the shield solves a Linear MPC (LMPC) problem rather than a Nonlinear MPC (NMPC) problem. This transforms a nonconvex optimization landscape into a convex one, enabling substantially faster solution times than current methods. Such a shield would be task-agnostic and applicable to high-degree-of-freedom systems, for which analytical modeling is intractable, and would not be limited to navigation scenarios as in [150], [151].

Predictive Coordination with Diffusion Models (Extending Chap. 4). In our multi-agent experiments, we observed that timeouts increased with team size due to local deadlocks. This occurs because the safety filter treats neighbors as static obstacles, ignoring their motion. A complementary direction is to augment the safety layer with *diffusion models* [152] for multi-agent trajectory forecasting. Unlike deterministic predic-

tors that output a single mean trajectory (often leading to freezing behavior in uncertain scenarios), diffusion models are generative probabilistic models that capture a multi-modal distribution over future behaviors. By integrating a diffusion-based forecaster into the distributed NMPC, agents could reason over multiple potential paths of their neighbors. Injecting these probabilistic forecasts as dynamic constraints would allow the MPC to plan around likely motions, negotiate right-of-way, and resolve complex crossings without explicit communication, thereby reducing deadlock rates in dense formations.

Synthetic Rollouts via Efficient Sequence Modeling (Extending Chap. 6). The action regulator (Chapter 6) eliminates the online solver but requires a cost critic trained on unsafe interactions. Collecting such data on physical hardware is dangerous. A promising direction to improve sample efficiency without increasing risk is to generate synthetic training data using *Mamba-based sequence models* [153]. While transformer-based models have shown promise, their quadratic computational complexity limits their ability to generate long-horizon rollouts quickly. Mamba, a selective state-space model (SSM), offers much of the modeling capacity of transformers while scaling linearly with sequence length. Unlike prior model-based RL methods such as Safe-Dreamer [124], which often employ simplified architectures that struggle with complex morphologies, we propose using Mamba to learn a fast, high-fidelity world model that can generate large numbers of unsafe trajectories (e.g., collisions or falls) in simulation. These synthetic rollouts would allow us to pre-train the cost critics to detect dangerous regions before the robot encounters them during online training. Since the critics serve only as a safety check, rather than a precise trajectory planner, the system is robust to moderate inaccuracies in the learned dynamics, making this a practical strategy for safe reinforcement learning and sim-to-real transfer.

Universal Safety Foundation Models. Finally, moving beyond task-specific safety, a compelling long-term direction is the development of *universal safety foundation models*. While current robot foundation models [154], [155] focus on generalizing task execution (e.g., "pick up the apple"), they often lack explicit safety guarantees. We propose scaling a learning-based safety shield into a general-purpose safety backbone. By pre-training a shield on diverse offline datasets; spanning navigation, manipulation, and locomotion, we could learn a universal learning-based shield that encodes fundamental physical constraints (e.g., "high velocity near obstacles is dangerous," "joint limits must be respected") across different robot morphologies. Such a foundation model would serve as a plug-and-play safety layer for new tasks, dramatically improving sample efficiency by relieving the RL agent from re-learning basic physics constraints from scratch in every new environment. This effectively decouples safety generalization from task specialization, paving the way for truly robust generalist robots.

Bibliography

[1] A. Zachariae, F. Plahl, Y. Tang, I. Mamaev, B. Hein, and C. Wurll, "Human-robot interactions in autonomous hospital transports," *Robotics and Autonomous Systems*, vol. 179, p. 104 755, 2024.

[2] A. Krnjaic, R. D. Steleac, J. D. Thomas, G. Papoudakis, L. Schäfer, A. W. K. To, K.-H. Lao, M. Cubuktepe, M. Haley, P. Börsting, et al., "Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 677–684.

[3] C. Cheng, J. Fu, H. Su, and L. Ren, "Recent advancements in agriculture robots: Benefits and challenges," *Machines*, vol. 11, no. 1, p. 48, 2023.

[4] P. Arm, G. Waibel, J. Preisig, T. Tuna, R. Zhou, V. Bickel, G. Ligeza, T. Miki, F. Kehl, H. Kolvenbach, et al., "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Science robotics*, vol. 8, no. 80, eade9548, 2023.

[5] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[6] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[7] I. Goodfellow, *Deep learning*, 2016.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[9] *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[12] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., "Grandmaster level in starcraft II using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[13] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 28 694–28 698.

[14] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, 2017.

[15] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous uav navigation with sparse rewards," *IEEE Internet of Things Journal*, 2020.

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

[17] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[18] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[19] C. Cutler and B. Ramaker, "Dynamic matrix control," in *A computer control algorithm. In joint automatic control conference*, vol. 17, 1980, p. 72.

[20] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[21] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *2017 American control conference (ACC)*, IEEE, 2017, pp. 5115–5120.

[22] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[23] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, "Tailoring solution accuracy for fast whole-body model predictive control of legged robots," *IEEE Robotics and Automation Letters*, 2024.

[24] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 559–576, 2005.

[25] J. B. Rawlings, D. Q. Mayne, M. Diehl, et al., *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2020, vol. 2.

[26] M. Zanon, V. Kungurtsev, and S. Gros, "Reinforcement learning based on real-time iteration nmpc," *ifac-Papersonline*, vol. 53, no. 2, pp. 5213–5218, 2020.

[27] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[28] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2016.

[29] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[30] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2018.

[31] J. DelPreto, J. I. Lipton, L. Sanneman, A. J. Fay, C. Fourie, C. Choi, and D. Rus, "Helping robots learn: A human-robot master-apprentice model using demonstrations via virtual reality teleoperation," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2020.

[32] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," *IFAC-PapersOnLine*, 2018.

[33] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, 2020.

[34] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[35] A. P. Vinod, S. Safaoui, A. Chakrabarty, R. Quirynen, N. Yoshikawa, and S. Di Cairano, "Safe multi-agent motion planning via filtered reinforcement learning," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022, pp. 7270–7276.

[36] M. Li, Y. Jie, Y. Kong, and H. Cheng, "Decentralized global connectivity maintenance for multi-robot navigation: A reinforcement learning approach," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022, pp. 8801–8807.

[37] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109 597, 2021.

[38] M. Dawood, S. Pan, N. Dengler, S. Zhou, A. P. Schoellig, and M. Bennewitz, "Safe multi-agent reinforcement learning for behavior-based cooperative navigation," *IEEE Robotics and Automation Letters*, 2025.

[39] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by pid lagrangian methods," in *International Conference on Machine Learning*, PMLR, 2020, pp. 9133–9143.

[40] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*, PMLR, 2017, pp. 22–31.

[41] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293–321, 1992.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations, ICLR*, 2016.

[43] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[44] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[47] D. P. Kingma and J. A. Ba, "A method for stochastic optimization. 3rd int," in *International Conference on Learning Representations, ICLR*, 2015, pp. 1–15.

[48] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[49] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[50] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear model predictive control: towards new challenging applications*, Springer, 2009, pp. 391–417.

[51] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[52] P. Agarwal, P. de Beaucorps, and R. de Charette, "Goal-constrained sparse reinforcement learning for end-to-end driving," *arXiv preprint arXiv:2103.09189*, 2021.

[53] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[54] M. Yi, X. Xu, Y. Zeng, and S. Jung, "Deep imitation reinforcement learning with expert demonstration data," *The Journal of Engineering*, 2018.

[55] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," International Foundation for Autonomous Agents and Multiagent Systems, AAMAS, 2020.

[56] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[57] F. Yao, C. Yang, X. Liu, and M. Zhang, "Experimental evaluation on depth control using improved model predictive control for autonomous underwater vehicle (auvs)," *Sensors*, 2018.

[58] B. B. Carlos, T. Sartor, A. Zanelli, G. Frison, W. Burgard, M. Diehl, and G. Oriolo, "An efficient real-time nmpc for quadrotor position control under communication time-delay," in *Proc. of the Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2020.

[59] M. Osman, M. W. Mehrez, S. Yang, S. Jeon, and W. Melek, "End-effector stabilization of a 10-dof mobile manipulator using nonlinear model predictive control," *IFAC-PapersOnLine*, 2020.

[60] M. Dawood, M. Abdelaziz, M. Ghoneima, and S. Hammad, "A nonlinear model predictive controller for autonomous driving," in *Proc. of the Intl. Conf. on Innovative Trends in Communication and Computer Engineering (ITCE)*, IEEE, 2020.

[61] M. W. Mehrez, G. K. Mann, and R. G. Gosine, "Stabilizing NMPC of wheeled mobile robots using open-source real-time software," in *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, IEEE, 2013.

[62] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., "Deep Q-learning from demonstrations," in *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 2018.

[63] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2022.

[64] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," in *International Conference on Learning Representations, ICLR*, 2022.

[65] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (*IROS*), IEEE, 2012.

[66] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni, "Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation* (*ICRA*), IEEE, 2018.

[67] G. Wang, M. Xin, W. Wu, Z. Liu, and H. Wang, "Learning of long-horizon sparse-reward robotic manipulator tasks with base controllers," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[68] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*, PMLR, 2013, pp. 1–9.

[69] G. Bellegarda and K. Byl, "An online training method for augmenting mpc with deep reinforcement learning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (*IROS*), IEEE, 2020.

[70] J. Shin, A. Hakobyan, M. Park, Y. Kim, G. Kim, and I. Yang, "Infusing model predictive control into meta-reinforcement learning for mobile robots in dynamic environments," *IEEE Robotics and Automation Letters* (*RA-L*), 2022.

[71] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[72] G. Turrisi, B. B. Carlos, M. Cefalo, V. Modugno, L. Lanari, and G. Oriolo, "Enforcing constraints over learned policies via nonlinear mpc: Application to the pendubot," *IFAC-PapersOnLine*, 2020.

[73] K. Lee, K. Saigol, and E. A. Theodorou, "Safe end-to-end imitation learning for model predictive control," *arXiv preprint arXiv:1803.10231*, 2018.

[74] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, *Acados: A modular open-source framework for fast embedded optimal control*, 2020. arXiv: `1910.13753 [math.OC]`.

[75] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, 2018.

[76] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems* (*IROS*), IEEE, 2004.

[77] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., "ROS: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[78] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of the Intl. Conf. on Machine Learning (ICML)*, PMLR, 2018.

[79] Y. Liu and G. Nejat, "Multirobot cooperative learning for semiautonomous control in urban search and rescue applications," *Journal of Field Robotics*, vol. 33, no. 4, pp. 512–536, 2016.

[80] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad, "Active perception based formation control for multiple aerial vehicles," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4491–4498, 2019.

[81] R. T. Fawcett, L. Amanzadeh, J. Kim, A. D. Ames, and K. A. Hamed, "Distributed data-driven predictive control for multi-agent collaborative legged locomotion," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2023, pp. 9924–9930.

[82] G.-P. Liu and S. Zhang, "A survey on formation control of small satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 440–457, 2018.

[83] D. Xu, X. Zhang, Z. Zhu, C. Chen, P. Yang, et al., "Behavior-based formation control of swarm robots," *mathematical Problems in Engineering*, vol. 2014, 2014.

[84] L. Quan, L. Yin, T. Zhang, M. Wang, R. Wang, S. Zhong, X. Zhou, Y. Cao, C. Xu, and F. Gao, "Robust and efficient trajectory planning for formation flight in dense environments," *IEEE Trans. on Robotics (TRO)*, 2023.

[85] Y. Yan, X. Li, X. Qiu, J. Qiu, J. Wang, Y. Wang, and Y. Shen, "Relative distributed formation and obstacle avoidance with multi-agent reinforcement learning," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022, pp. 1661–1667.

[86] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2358–2372, 2020.

[87] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, "Graph policy gradients for large scale robot control," in *Proceedings of the Conference on Robot Learning*, 2019, pp. 823–834.

[88] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, PMLR, 2022, pp. 91–100.

[89] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 23–30.

[90] Z. Zhang, S. Han, J. Wang, and F. Miao, "Spatial-temporal-aware safe multi-agent reinforcement learning of connected autonomous vehicles in challenging scenarios," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023, pp. 5574–5580.

[91] Z. Sheebaelhamd, K. Zisis, A. Nisioti, D. Gkouletsos, D. Pavllo, and J. Kohler, "Safe deep reinforcement learning for multi-agent systems with continuous action spaces," arXiv:03952, 2021.

[92] I. ElSayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng, "Safe multi-agent reinforcement learning via shielding," *arXiv preprint arXiv:2101.11196*, 2021.

[93] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," arXiv:04043, 2019.

[94] P. Zhang, G. Chen, Y. Li, and W. Dong, "Agile formation control of drone flocking enhanced with active vision-based relative localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 6359–6366, 2022.

[95] T. Xu, J. Liu, Z. Zhang, G. Chen, D. Cui, and H. Li, "Distributed MPC for trajectory tracking and formation control of multi-uavs with leader-follower structure," *IEEE Access*, pp. 1–1, 2023. DOI: 10.1109/ACCESS.2023.3329232.

[96] S. Park and S.-M. Lee, "Formation reconfiguration control with collision avoidance of nonholonomic mobile robots," *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[97] S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme, "Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, 2022, pp. 576–586.

[98] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "Amswarmx: Safe swarm coordination in complex environments via implicit non-convex decomposition of the obstacle-free space," 2024.

[99] L. Quan, L. Yin, C. Xu, and F. Gao, "Distributed swarm trajectory optimization for formation flight in dense environments," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2022, pp. 4979–4985.

[100] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022, pp. 24 611–24 624.

[101] J. Xie, R. Zhou, Y. Liu, J. Luo, S. Xie, Y. Peng, and H. Pu, "Reinforcement-learning-based asynchronous formation control scheme for multiple unmanned surface vehicles," *Applied Sciences*, vol. 11, no. 2, p. 546, 2021.

[102] X. Zhou, P. Wu, H. Zhang, W. Guo, and Y. Liu, "Learn to navigate: Cooperative path planning for unmanned surface vehicles using deep reinforcement learning," *IEEE Access*, vol. 7, pp. 165 262–165 278, 2019.

[103] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10 980–10 992, 2023.

[104] Z. Cai, H. Cao, W. Lu, L. Zhang, and H. Xiong, "Safe multi-agent reinforcement learning through decentralized multiple control barrier functions," *arXiv preprint arXiv:2103.12553*, 2021.

[105] 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)*, IEEE, 2019, pp. 3420–3431.

[106] W. Zhang, O. Bastani, and V. Kumar, "Mamps: Safe multi-agent reinforcement learning via model predictive shielding," *arXiv preprint arXiv:1910.12639*, 2019.

[107] X. Zhang, Y. Peng, W. Pan, X. Xu, and H. Xie, "Barrier function-based safe reinforcement learning for formation control of mobile robots," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022, pp. 5532–5538.

[108] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[109] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[110] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[111] F. P. Bejarano, L. Brunke, and A. P. Schoellig, "Multi-step model predictive safety filters: Reducing chattering by increasing the prediction horizon," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, IEEE, 2023, pp. 4723–4730.

[112] M. Dawood, N. Dengler, J. de Heuvel, and M. Bennewitz, "Handling sparse rewards in reinforcement learning using model predictive control," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2023, pp. 879–885.

[113] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[114] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking reinforcement learning techniques for autonomous navigation," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2023, pp. 9224–9230.

[115] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Robotics: Science and Systems (RSS)*, 2024.

[116] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *International Conference on Learning Representations*, 2019.

[117] A. Agha, B. Kayalibay, A. Mirchev, P. van der Smagt, and J. Bayer, "Exploring under constraints with model-based actor-critic and safety filters," in *8th Annual Conference on Robot Learning*, 2024.

[118] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[119] Z. Zhang, S. Han, J. Wang, and F. Miao, "Spatial-temporal-aware safe multi-agent reinforcement learning of connected autonomous vehicles in challenging scenarios," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023, pp. 5574–5580.

[120] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," *arXiv preprint arXiv:1910.01708*, vol. 7, no. 1, p. 2, 2019.

[121] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[122] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar, "Sauté rl: Almost surely safe reinforcement learning using state augmentation," in *International Conference on Machine Learning*, PMLR, 2022, pp. 20 423–20 443.

[123] S. Carr, N. Jansen, S. Junges, and U. Topcu, "Safe reinforcement learning via shielding under partial observability," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, pp. 14 748–14 756.

[124] W. Huang, J. Ji, B. Zhang, C. Xia, and Y. Yang, "Safedreamer: Safe reinforcement learning with world models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: `https://openreview.net/forum?id=tsE5HLYtYg`.

[125] Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang, "Omnisafe: An infrastructure for accelerating safe reinforcement learning research," *arXiv preprint arXiv:2305.09304*, 2023.

[126] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[127] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, `http://pybullet.org`, 2016.

[128] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, IEEE, 2017, pp. 3389–3396.

[129] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on robot learning*, PMLR, 2023, pp. 2226–2240.

[130] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

[131] B. Dai, R. Khorrambakht, P. Krishnamurthy, V. Gonçalves, A. Tzes, and F. Khorrami, "Safe navigation and obstacle avoidance using differentiable optimization based control barrier functions," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5376–5383, 2023.

[132] A. Banerjee, K. Rahmani, J. Biswas, and I. Dillig, "Dynamic model predictive shielding for provably safe reinforcement learning," *arXiv preprint arXiv:2405.13863*, 2024.

[133] A. Agha, B. Kayalibay, A. Mirchev, P. van der Smagt, and J. Bayer, "Exploring under constraints with model-based actor-critic and safety filters," in *8th Annual Conference on Robot Learning*, 2024.

[134] A. Sootla, A. Cowen-Rivers, J. Wang, and H. Bou Ammar, "Enhancing safe exploration using safety state augmentation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 464–34 477, 2022.

[135] A. Navon, A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya, "Multi-task learning as a bargaining game," *arXiv preprint arXiv:2202.01017*, 2022.

[136] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety gymnasium: A unified safe reinforcement learning benchmark," *Advances in Neural Information Processing Systems*, vol. 36, pp. 18 964–18 993, 2023.

[137] H. Tian, H. Hamedmoghadam, R. Shorten, and P. Ferraro, "Reinforcement learning with adaptive regularization for safe control of critical systems," *arXiv preprint arXiv:2404.15199*, 2024.

[138] H. Yu, W. Xu, and H. Zhang, "Towards safe reinforcement learning with a safety editor policy," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2608–2621, 2022.

[139] M. Ganai, Z. Gong, C. Yu, S. Herbert, and S. Gao, "Iterative reachability estimation for safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 69 764–69 797, 2023.

[140] D. Kim, T. Cho, S. Han, H. Chung, K. Lee, and S. Oh, "Spectral-risk safe reinforcement learning with convergence guarantees," *Advances in Neural Information Processing Systems*, 2024.

[141] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International Conference on Machine Learning*, PMLR, 2015, pp. 997–1005.

[142] Q. Zhang, S. Leng, X. Ma, Q. Liu, X. Wang, B. Liang, Y. Liu, and J. Yang, "CVaR-constrained policy optimization for safe reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[143] Y. Yao, Z. Liu, Z. Cen, J. Zhu, W. Yu, T. Zhang, and D. Zhao, "Constraint-conditioned policy optimization for versatile safe reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 12 555–12 568, 2023.

[144] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[145] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[146] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[147] A. W. Goodall and F. Belardinelli, "Leveraging approximate model-based shielding for probabilistic safety guarantees in continuous environments," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024.

[148] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[149] S. Khorshidi, M. Dawood, and M. Bennewitz, "Centroidal state estimation based on the koopman embedding for dynamic legged locomotion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 12 832–12 839.

[150] 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 AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 3387–3395.

[151] J. Kim, Y. Han, H. Ravichandar, and S. Ha, "Learning koopman dynamics for safe legged locomotion with reinforcement learning-based controller," *arXiv preprint arXiv:2409.14736*, 2024.

[152] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[153] A. Gu and T. Dao, *Mamba: Linear-time sequence modeling with selective state spaces*, arXiv:2312.00752, 2024.

[154] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al., "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[155] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, W. Ayza, D. Debidatta, K. Liu, and W. Huang, "Palm-e: An embodied multimodal language model," in *International Conference on Machine Learning (ICML)*, 2023.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

CBF	Control Barrier Function
CMDP	Constrained Markov Decision Process
CNN	Convolutional Neural Network
CTDE	Centralized Training with Decentralized Execution
DDP	Differential Dynamic Programming
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DOF	Degrees of Freedom
DQN	Deep Q-Network
GCN	Graph Convolutional Network
GPS	Guided Policy Search
HER	Hindsight Experience Replay
MARL	Multi-Agent Reinforcement Learning
MASAC	Multi-Agent Soft Actor-Critic
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MDP	Markov Decision Process
MIMO	Multi-Input Multi-Output
MLP	Multilayer Perceptron
MPC	Model Predictive Control
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
PID	Proportional-Integral-Derivative
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
RC	Return-to-Cost
RL	Reinforcement Learning
ROS	Robot Operating System
SAC	Soft Actor-Critic
SGD	Stochastic Gradient Descent
TB	Turtlebot
TD3	Twin Delayed Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Optimization