Towards Trustworthy Autonomy in Uncertain Environments: Synthesis of Verified Safe and Performant Control Policies for Autonomous Systems

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

by

Manan Tayal



to

Center for CYBER PHYSICAL SYSTEMS
INDIAN INSTITUTE OF SCIENCE
BENGALURU
INDIA - 560012

ACKNOWLEDGEMENT

Research is often presented as a sequence of ideas, theorems, and experiments, but behind every piece of work lies a network of people whose conversations, encouragement, and guidance shape the journey. Looking back on my Ph.D., I realize that this thesis is not only a record of research results, but also a reflection of the many individuals who influenced and supported me along the way. I am deeply grateful to all of them.

My journey into safe learning-based control began with a simple question: how can we design autonomous systems that learn from data while still providing rigorous safety guarantees? Much of my work during the Ph.D. revolved around exploring this question from multiple perspectives at the intersection of control theory, machine learning, and robotics. The ideas presented in this thesis were shaped not only by formal analysis and experimentation, but also by many conversations with mentors, collaborators, and colleagues who helped refine the questions I was trying to answer.

First and foremost, I would like to thank my advisors, **Prof. Shishir Kolathaya** and **Prof. Pushpak Jagtap**, for their guidance and support throughout my doctoral journey. Prof. Kolathaya provided me with the freedom to explore diverse research directions and encouraged me to shape and refine my own research questions, particularly during the early stages of my Ph.D. The independence I was given taught me how to navigate research problems on my own and played an important role in shaping my development as a researcher. Prof. Jagtap had a significant influence on how I approach research writing and dissemination. His emphasis on presenting work clearly and submitting to the right conferences and journals greatly improved how I communicate research. He also emphasized the importance of collaborations and helped connect me with researchers whose perspectives enriched this work.

I am deeply grateful to the collaborators who influenced many of the ideas in this thesis. **Prof. Somil Bansal** introduced me to the deep connections between reachability analysis and learning-based safety, and many ideas in this thesis grew out of discussions with him. I am also grateful to **Prof. Andrew Clark** and his student **Dr. Hongchao Zhang**. Their perspectives from formal methods brought additional rigor to several theoretical aspects of this work, and our discussions were invaluable in shaping some of the ideas presented here. I would also like to thank **Prof. Jishnu Keshavan** and **Prof. Ravi Prakash** for providing opportunities to test some of these ideas on real robotic platforms, particularly drones and manipulation systems. Working with their student **Rajpal Singh** was an equally rewarding experience.

During my Ph.D., I had the opportunity to work closely with several talented students. Mentoring them was one of the most rewarding aspects of my doctoral journey, and in many ways I learned

as much from them as they might have learned from me. **Aditya Singh**, **Karthik Rajgopal**, **G. V. S. Mothish**, **Ganga Nair**, and **Ravi Kola** all contributed energy, curiosity, and persistence to the projects we worked on together. Some of these collaborations focused on safety-critical control, while others grew out of earlier explorations on the control and design of bipedal robots. Although bipedal locomotion ultimately lies outside the scope of this thesis, those early efforts were formative and helped shape the trajectory of my research.

The day-to-day experience of a Ph.D. is defined not only by research progress but also by the people with whom one shares the lab. I am grateful to my labmates **Aman Singh**, **Aditya Shirwatkar**, **Vamshi Kumar Kurva**, **Aastha Mishra**, and **Prakrut Kotecha** for making this journey intellectually stimulating and enjoyable. Discussions with Aman on design and control were always insightful, and a cup of chai with him often helped put research stress into perspective. Our game nights, weekend cricket matches, and long conversations about cricket made the Ph.D. years memorable. I also enjoyed many stimulating discussions with Prakrut on world models and learning-based approaches to control, and with Aastha on the challenges of co-optimizing design and control. Aditya Shirwatkar's deep familiarity with the legged robotics literature frequently saved us from rediscovering old ideas. His famous remark, *"Ye to pehle hi ho gaya hai"* (It is already done), eventually became a running joke in our group.

I would also like to thank my batchmates **Alokendu Mazumder** and **Piyush Tiwary**, with whom I shared many of the early milestones of the Ph.D. journey. Taking courses together such as *Reinforcement Learning*, *Computational Methods of Optimization*, *PRNN* and *StoMA* made those early semesters far more manageable. Working through assignments, preparing for exams, and discussing ideas together helped build friendships that I deeply value.

A special thanks goes to **Preethi Ma'am**, Prof. Kolathaya's administrative assistant. Her help with conference travel, equipment purchases, reimbursements, and countless other administrative tasks made the practical aspects of research significantly smoother. Her efficiency and willingness to help ensured that administrative hurdles rarely became obstacles.

I gratefully acknowledge the financial and institutional support that made this research possible. I thank the **Prime Minister's Research Fellowship (PMRF)** for supporting my doctoral studies. I am also grateful to **ARTPARK** and the **Wipro IISc Research Innovation Network (WIRIN)** for supporting various aspects of this research, including computational resources, equipment procurement, and conference travel.

Finally, I thank my family. None of this would have been possible without their unwavering support. My parents, **Mohan Tayal** and **Madhvi Tayal**, have always encouraged me to pursue

my interests with dedication and integrity. Their quiet sacrifices and constant belief in me made this journey possible. My brother, **Mumuksh Tayal**, has been a constant presence throughout these years, often serving as a sounding board, sometimes as a collaborator and co-author, and always as someone who reminds me not to take myself too seriously. This thesis is as much theirs as it is mine.

Looking back, this Ph.D. has been as much a journey of personal growth as it has been one of academic discovery. If the ideas in this thesis contribute even a small step toward building safer and more reliable autonomous systems, it will be because of the many people who guided, challenged, and supported me along the way.

Manan Tayal
Bengaluru, India

ABSTRACT

Autonomous systems are increasingly deployed in uncertain, dynamic, and safety-critical environments where reliability is essential. Trustworthy autonomy in such settings rests on two fundamental pillars: safety, ensuring that catastrophic states are never reached, and performance, ensuring that control objectives are achieved efficiently. Classical control theory provides rigorous safety guarantees through tools such as Control Barrier Functions and Hamilton–Jacobi reachability analysis. However, these approaches rely on accurate system models and face significant scalability limitations in high-dimensional or perception-driven settings. Learning-based control methods like Reinforcement Learning, on the other hand, scales to complex environments and operate under model uncertainty, yet they typically lack any formal assurances. Bridging this divide is central to enabling trustworthy autonomous systems.

This thesis develops a unified framework for learning verified safe control policies under uncertainty and progressively extending them toward safety–performance co-optimization in increasingly complex systems. The central premise is that safety certificates can be parameterized by neural networks, synthesized from finite datasets, and verified or statistically calibrated to retain rigorous guarantees.

The thesis begins by addressing safety verification for deterministic and stochastic control-affine systems. Neural Control Barrier Functions are learned from sampled data and certified using Lipschitz-based bounds and verification-in-the-loop synthesis. To reduce conservatism and remove dependence on global smoothness assumptions, split conformal prediction is incorporated to obtain distribution-free, finite-sample safety guarantees for learned barrier functions.

Building on verified safety, the framework then moves beyond local forward invariance. Collision-cone barrier formulations incorporate geometric look-ahead, and Value-Guided Control Barrier Functions unify reachability analysis with offline reinforcement learning to approximate long-horizon safe sets from data without requiring explicit system models.

The thesis next advances toward joint safety and performance co-optimization as a single objective. State-constrained optimal control is reformulated as a Hamilton–Jacobi–Bellman Partial Differential Equation and solved using physics-informed neural networks, enabling the synthesis of control policies that are both safe and near-optimal. Conformal calibration provides finite-sample reliability guarantees, and decentralized formulations allow scalability to high-dimensional and multi-agent systems.

Finally, the framework is extended from state-based control to perception-driven autonomy by learning barrier functions in latent spaces derived from raw visual observations. This enables certified visuomotor control without explicit state estimation, thereby scaling safety guarantees to high-dimensional sensory inputs.

Collectively, this thesis demonstrates that safe control policies can be learned from finite data, formally verified under uncertainty, and systematically extended toward safety–performance co-optimization in high-dimensional and multi-agent settings. By progressing from safety verification to scalable, perception-driven control, this work contributes toward a principled foundation for trustworthy autonomy in uncertain environments.

Keywords:

Neural Safety Certificates, Control Barrier Functions, Hamilton–Jacobi Reachability, Safe Reinforcement Learning, Conformal Prediction, Physics-Informed Learning, Multi-Agent Systems, Visuomotor Control, Trustworthy Autonomy

Contents

List of Figures **xiv**

List of Tables **xxv**

Abbreviations xxix

Symbols xxix

1 Introduction **1**

1.1 Formal Safety and Scalable Learning 2

1.2 From Analytical to Learned Safety 3

1.3 Central Claim 4

1.4 Thesis Overview 5

2 Background and Preliminaries **9**

2.1 Dynamical Systems 9

2.2 Safety and Invariance 12

2.3 Control Barrier Functions 13

2.4 Stochastic Systems and Barrier Conditions 16

2.5 Robust Control Barrier Functions 19

2.6 Dynamic Programing and HJ Formulations . . . 21
2.7 Physics-Informed Neural Networks . . . 27
2.8 Reinforcement Learning and Markov Decision Processes . . . 31
2.9 Conformal Prediction for Safety and Verification . . . 37
2.10 Latent Representations and Visuomotor Control . . . 38
2.11 Summary of Background . . . 40

I Neural Control Barrier Functions 42

3 Lipschitz-Verified Neural Control Barrier Functions 43

3.1 Related Works . . . 43
3.2 Contributions of this Chapter . . . 44
3.3 Problem Formulation . . . 45
3.4 Method . . . 48
3.5 Simulations . . . 52
3.6 Chapter Summary . . . 57

4 Conformal Neural CBFs 59

4.1 Related Works . . . 59
4.2 Contributions of this Chapter . . . 60
4.3 Problem Formulation . . . 60
4.4 Conformal Neural CBF . . . 62
4.5 Experiments . . . 66
4.6 Chapter Summary . . . 70

5 Verification-in-the-Loop Synthesis of Stochastic Neural CBFs **72**

5.1 Related Works . . . 72

5.2 Contributions of this Chapter . . . 74

5.3 Preliminaries . . . 74

5.4 Smooth Stochastic Neural Control Barrier Functions . . . 81

5.5 Rectified Linear Unit Stochastic Control Barrier Functions . . . 92

5.6 Experiments . . . 102

5.7 Chapter Summary . . . 108

II Towards Planning-Aware Safety **111**

6 Collision Cone CBFs: Planning-Aware Safety for Robotics **112**

6.1 Related Works . . . 112

6.2 Contributions of this Chapter . . . 113

6.3 Preliminaries . . . 113

6.4 Collision Cone Control Barrier Function . . . 118

6.5 Results . . . 128

6.6 Chapter Summary . . . 133

7 Value-Guided Offline Neural CBF Synthesis **138**

7.1 Related Work . . . 138

7.2 Contributions of this Chapter . . . 139

7.3 Background and Problem Setup . . . 140

7.4 Methodology . . . 143
7.5 Experiments . . . 147
7.6 Chapter Summary . . . 153

III Bridging Safety with Performance **155**

8 Co-optimizing Safety and Performance via Physics-Informed ML **156**
8.1 Related Works . . . 156
8.2 Contributions of this Chapter . . . 158
8.3 Problem Setup . . . 158
8.4 Safe and Optimal Control with Perfect Information and no Disturbances . . . 162
8.5 Safe and Optimal Control with Imperfect Information and Disturbances . . . 176
8.6 Hardware Experiments . . . 186
8.7 Chapter Summary . . . 189

9 Scaling Safety–Performance Co-optimization to Multi-Agent Systems **190**
9.1 Related Works . . . 190
9.2 Contributions of this Chapter . . . 191
9.3 Problem Setup . . . 192
9.4 Methodology . . . 195
9.5 Experiments . . . 200
9.6 Chapter Summary . . . 205

IV Safety in Latent Spaces **206**

10 Safe Visuomotor Control with Barrier Certificates **207**

10.1 Related Works . . . 207

10.2 Contributions of this Chapter . . . 208

10.3 Preliminaries and Problem Formulation . . . 209

10.4 Semi-Supervised Policy Learning Framework . . . 212

10.5 Case Studies . . . 217

10.6 Chapter Summary . . . 221

11 Discussion and Future Directions **222**

11.1 Summary of Contributions . . . 222

11.2 Scope of Guarantees and Assumptions Across Chapters . . . 223

11.3 Future Challenges and Opportunities . . . 224

11.4 Closing Remarks . . . 229

Appendices **232**

A Appendix of Chapter 6 **232**

A.1 Proofs . . . 232

B Appendix of Chapter 3 **238**

B.1 Proofs . . . 238

C Appendix of Chapter 4 **241**

C.1 Proofs . . . 241

C.2 Relationship between α, β, and ϵ . . . 243

C.3 Experimental Details . . . 244

C.4 Implementation Details of the Algorithms . . . 248

D Appendix of Chapter 7 **251**

D.1 Proofs 251

D.2 Description of the Experiments 255

D.3 Ablation Studies 258

D.4 Additional Experiments on Boat Navigation 262

D.5 Experimental Details 263

E Appendix of Chapter 8 **267**

E.1 Proofs 267

E.2 Additional Details the systems in the experiments 274

E.3 Implementation Details of the Algorithms 280

Bibliography **285**

List of Figures

1.1 Classical control theory (left) provides rigorous safety guarantees but requires accurate models and scales poorly; machine learning (right) offers flexibility and scalability without formal assurances. This thesis develops methods at the intersection, combining learning-based safety and optimality tools with formal guarantees. 3

1.2 From analytical to learned safety. Classical safety synthesis (top, grey) requires known dynamics and hand-crafted barrier functions verified analytically. This thesis (bottom, teal) learns neural barrier and value functions from data and verifies or calibrates them via Lipschitz bounds [165], conformal prediction [158], or verification-in-the-loop synthesis [197]. 4

1.3 Thesis architecture. The dissertation progresses through four parts: from learning and verifying neural safety functions (Part I), through long-horizon and data-driven safety (Part II), to joint safety–performance optimization (Part III), and finally certified visuomotor control from raw perception (Part IV). Arrows indicate the conceptual progression. 5

3.1 Plot of $B_\theta(x)$ over X. The horizontal and vertical axes are the pendulum angle θ [rad] and angular velocity $\dot{\theta}$ [rad/s], respectively. Blue and red regions denote the safe region ($B \geq 0$) and the unsafe region ($B < 0$), respectively. The initial safety region boundary and unsafe region boundary is denoted by their respective regions marked with coloured oblique lines. The black line denotes the zero-level contour of the barrier function. We note that the framework does not require square or box-constrained safe/unsafe sets; such sets are used here purely for ease of specification, and arbitrary (e.g., polytopic or nonconvex) sets can be handled identically since the method relies only on sampling and Lipschitz bounds. 53

3.2 Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed NCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian. 55

3.3 The left and right figures show the unsafe region and the zero-level sets of the NCBF B trained by baseline and the proposed method, respectively. Both zero-level sets (in yellow) do not overlap with the unsafe region in red color. The NCBF trained by the baseline ensures a guaranteed safe subset over the state space with the ratio 11.8% only, whereas the proposed method ensures a guaranteed safe subset with a ratio up to 77.85%. 56

3.4 Learned cart-pole NCBF evaluated over the velocity grid. 57

4.1 **Overview of the proposed approach**: The proposed methodology consists of **three main steps**. In the **first step**, the NCBF, denoted by B_θ, is trained using an initial safety margin parameter $\psi = 0$. **In the second step**, the conformal score $\hat{q}$ is computed based on the policy induced by the learned NCBF. **In the third step**, this conformal score is then used to update the safety margin ψ, which is subsequently employed to retrain the NCBF. This iterative process is repeated until the conformal score converges to zero, thereby ensuring the desired probabilistic safety guarantee. 61

4.2 Visualization of learned NCBFs for autonomous ground vehicle at $\phi = 0$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (calibrated at a 99.9% marginal coverage level), and (Right) Verified NCBF [165]. Bold black ovals denote the zero-level sets. In this visualization, both **CP-NCBF and Verified NCBF avoid overlapping the unsafe region, while the Unverified NCBF does not**; additionally, CP-NCBF yields **a larger safe region** than the Verified NCBF. 67

4.3 Visualization of learned NCBFs for the aerial vehicle at $\phi, \dot{x}_1, \dot{x}_2, \dot{\phi} = 0$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (with **99.5% safety guarantee**), and (Right) CP-NCBF (with **99.9% safety guarantee**). Bold black lines denote the zero-level sets. As the desired **safety guarantees become more stringent**, the learned NCBF **exhibits increased conservatism**. This behavior highlights the flexibility of our proposed framework in enabling a **systematic trade-off between safety and performance, tailored to the specific requirements of the system.** 68

4.4 This figure presents a comparative analysis of the empirical safety rates achieved by all evaluated methods. **The observed safety rates align closely with the corresponding theoretical guarantees**, thereby providing empirical validation of the reliability and robustness of the proposed framework. These findings highlight the potential of our approach for **deployment in real-world safety-critical applications**. We also note that **VNCBF could not be computed** for the geo-fencing and quadruped navigation tasks due to its **reliance on grid-based sampling, which is computationally feasible only for systems up to 5 dimensions**. This limitation further illustrates the **scalability advantage of our method, which remains tractable and effective even in high-dimensional settings.** 69

5.1 Workflow of the synthesis with verification in the loop. 88

5.2 This figure presents the experimental results on the inverted pendulum system. Fig. 5.2a visualizes of $B(\mathbf{x})$ over $\mathcal{X}$. Blue and red regions denote the safe region ($B \geq -\psi^*$) and the unsafe region ($B < \psi^*$), respectively. The initial safety region boundary and unsafe region boundary is denoted by black boxes. We observe that the boundary of trained SNCBF (black dots) successfully separate the unsafe and safe region. Fig. 5.2b shows the 3D plot of $B(\mathbf{x})$ over $\mathcal{X}$. Fig. 5.2c presents trajectories initiating inside the safe set following SNCBF-QP, following different reference controllers 105

5.3 Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed SNCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian. 105

5.4 Comparison of coverage and training time across Baseline, Smooth and ReLU SNCBF Synthesis. Error bars indicate standard deviations across 5 seeds. 107

6.1 Schematic of Unicycle (left); Bicycle model (right); Quadrotor model (down). 114

6.2 Construction of collision cone for an elliptical obstacle (where c_1 & c_2 are the major & minor axis length respectively) considering the ego-vehicle's dimensions (width: w). 119

6.3 **3D CBF** candidate: The dimensions of the obstacle are comparable to each other, it can be assumed as a sphere 121

6.4 **Projection CBF** candidate: One of the dimensions, of the obstacle, is bigger than the other dimensions, it can be assumed as a cylinder. 124

6.5 Comparison of HO-CBF with C3BF. Here we are trying to compare the ϕ' and ϕ obtained from the two CBF formulations. It can be observed that ϕ' (pink cone) is dependent on v_{rel}, while ϕ (yellow cone) is a constant, for a perticular distance from the obstacle. 126

6.6 Test setups: TurtleBot3 Burger, Stoch-Jeep, and Crazyflie 2.1, respectively. 127

6.7 Experimental Setup for TurtleBot Burger, Stoch-Jeep and Crazyflie-2.1. 128

6.8 Interaction of TurtleBot3 (Unicycle Model) with (a) Static Obstacle; (b) Dynamic Obstacle (Velocity towards the agent); (c) Dynamic Obstacle (Overtaking); (d) Dynamic Obstacle (Velocity perpendicular to & towards the agent). 130

6.9 Interaction of Stoch-Jeep (Bicycle Model) with (a) Static Obstacle; (b) Dynamic Obstacle (Velocity towards the agent); (c) Dynamic Obstacle (Overtaking); (d) Dynamic Obstacle (Velocity perpendicular to & towards the agent). 131

6.10 Interaction of Crazyflie 2.1 (Quadrotor) with (a) Static Obstacle; (b) Long (Static) Obstacle; (c) Dynamic Obstacle (Velocity towards the agent). 132

6.11 Robustness in scenarios with multiple obstacles: (a) TurtleBot3 (Unicycle Model) [Dynamic Obstacles]; (b) Stoch-Jeep (Bicycle Model) [Dynamic Obstacles]; (c) Quadrotor (Crazyflie 2.1) - Long (Static) Obstacles; (d) Quadrotor (Crazyflie 2.1) - Spherical (Static) Obstacles 134

6.12 Comparison of thrusts between Collision Cone CBF and Higher Order CBF 135

6.13 Experiments with a dynamic obstacle (unicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 135

6.14 Experiments with multiple obstacles (unicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 135

6.15 Experiments with dynamic obstacles (bicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 135

6.16 Experiments with multiple obstacles (bicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 136

6.17 Experiments with a dynamic obstacle (quadrotor): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 136

6.18 Experiments with multiple obstacles (quadrotor): a) Traced Path, Evolution of b) linear velocity, c) control inputs. 136

7.1 **Framework Overview:** *(Left)*: We learn value guided barrier function with reachability based bellman equation for learning optimal safe region and apply expectile regression for OOD case handling, when learning from Offline Dataset. *(Right)*: Inferencing with CBF-QP using learned barrier function as valid CBF to rollout safe step-wise actions, with any reference controller. 145

7.2 **Illustrations of the test environments:** (*Top-Left*) environment illustrates Autonomous Ground Vehicle with an obstacle and a goal point. (*MuJoCo Env*) **Red** sphere denotes unsafe state. 148

7.3 Results plot for all the Mujoco based Safety Gymnasium [77] environments. Points towards left (←) are more safe than those on right (→). Evaluated over 500 episodes and 5 seed values. 151

7.4 **Qualitative comparison of Hopper rollouts with and without V-OCBF.** *(Top):* When using V-OCBF, the agent adapts its gait to keep its forward velocity within the allowable safety threshold, demonstrating how the learned barrier actively regulates unsafe accelerations. *(Bottom):* Instantaneous velocities across episodes for both methods. The left axis reports the velocity profile, while the right axis shows the fraction of episodes that violate the safety threshold at each timestep. V-OCBF substantially reduces violation frequency, illustrating its ability to enforce safety constraints during deployment. Additional qualitative results for other environments are provided in Appendix D.2.2. 151

8.1 **Overview of the proposed approach**: The methodology is organized into four steps. The **first step** involves training the auxiliary value function, $\hat{V}_\theta$, using a physics-informed machine learning framework. The **second step** applies a conformal prediction approach for safety verification of the learned $\hat{V}_\theta$. In the **third step**, the final value function V_θ and the optimal safe and performant policy π_θ are inferred. The **fourth step** quantifies the performance of V_θ through a second conformal prediction procedure. 164

8.2 This figure presents a comparative study between all the methods based on our evaluation metrics. The top plot illustrates the **mean percentage increase in cumulative cost** relative to our method for each baseline, demonstrating that our approach consistently incurs lower costs, with the gap widening as system complexity grows. The bottom plot depicts the **safety rates**, showing that our method maintains a 100% safety rate, while baselines that encourage safety rather than enforcing it (like MPPI and C-SAC) achieve lower rates. MPPI-CBF also attains 100% safety but at the expense of performance. Overall, our method uniquely **balances both safety and performance**, whereas the baselines compromise on at least one aspect. 168

8.3 Trajectories from two distinct initial states are shown, with dark grey circles representing obstacles and the green dot indicating the goal at $[1.5, 0]^T$. Notably, our method is the **only one** that **successfully approaches the goal** while **adhering to safety constraints**. 170

8.4 Trajectories from two distinct initial states are depicted, with dark grey circles representing obstacles and purple trajectories indicating the evader's path, with arrows showing its direction of motion. Our method successfully **tracks the evader** while **avoiding collisions**, whereas all other methods either fail to maintain safety, struggle to track the evader or both 171

8.5 Trajectories of the **receding horizon policy** for the pursuer tracking an evader over a 6-second horizon, while the value function is trained over a 1-second horizon. The results demonstrate that the pursuer **successfully tracks** the evader while ensuring safety, even when initialized near the obstacle. This highlights the effectiveness of the proposed approach in **jointly optimizing safety and performance for long-horizon tasks.** 173

8.6 Snapshots of multi-agent navigation trajectories at different times using the proposed method. Agents are represented as circles with radius R, indicating the minimum safe distance they must maintain from each other. Smaller dots mark their respective goals. The trajectories show that agents proactively **maintain long-horizon safety** by adjusting their paths to avoid close encounters, rather than enforcing safety reactively, which could lead to suboptimal behaviors. Finally, the agents **reach their respective goals within the specified time horizon**. 174

8.7 This figure presents a comparative analysis of all methods based on online and offline computation time evaluated on the same computing machine. The top plot illustrates the **offline computation time** for our method and the baselines. Since our method and SAC-Lag involve training value functions, they incur higher offline computation costs, whereas MPPI-based methods require no offline training. The bottom plot depicts the **online computation time**, demonstrating that our method and SAC-Lag have minimal online computation requirements, whereas MPPI-based methods exhibit significantly higher online computational costs. 175

8.8 Trajectories from two different initial conditions are shown for value functions trained under varying disturbance levels and evaluated in simulation with different disturbance magnitudes. (a) Evaluation disturbance $d = 0.0$. (b) Evaluation disturbance $d = 0.1$. (c) Evaluation disturbance $d = 0.2$. The results indicate that policies trained with disturbance levels greater than or equal to those encountered during evaluation exhibit improved safety, highlighting the importance of learning robust value functions for maintaining safety under uncertainty. Additionally, policies trained with disturbance levels exceeding those present in the environment exhibit increased conservatism, although the performance degradation remains modest. These results demonstrate that the proposed framework can synthesize safe policies for uncertain environments without substantial performance loss. 181

8.9 Trajectories from two distinct initial conditions are shown for value functions trained with and without disturbance in the evader model. The policy derived from the robust value function effectively adapts to the evader's erratic behavior arising from model uncertainty and successfully maintains tracking. In contrast, the non-robust policy exhibits limited adaptation, resulting in less effective tracking and consequently higher tracking costs. 185

8.10 Hardware experiment setup for the pursuer-evader tracking task. Two customized TurtleBot3 Burger platforms fitted with mecanum wheels are used: the blue-topped robot (pursuer, unicycle dynamics) tracks the yellow-topped robot (evader, point-mass dynamics) while avoiding box-shaped obstacles. The dashed lines indicate representative trajectories of the pursuer (blue) and evader (yellow). Global localization is provided by the PhaseSpace™ motion capture system, and policy inference is performed on an offboard PC that sends control commands to both robots in real time. 187

8.11 This figure presents two representative trajectories obtained from the hardware experiments. The pursuer adapts to the evader's erratic motion and maintains successful tracking, demonstrating the robustness of the proposed approach to disturbances encountered during sim-to-real transfer and its potential for real-world deployment. 188

9.1 We propose **MAD-PINN** – a framework for safe and optimal multi-agent control. MAD-PINN is divided into two phases: (1) **Centralized training**, where we learn the auxiliary epigraph-based value function $\hat{V}_\theta$ using a boundary-aware PINN, and the pairwise safety value function V_s via DeepReach; (2) **Decentralized deployment**, where each agent selects its safety-critical neighbours using V_s and executes the policy in a receding-horizon manner using $\hat{V}_\theta$. This design enables tractable training, adaptive neighbour selection, and scalable execution in large multi-agent systems. 196

9.2 Comparison of cumulative costs and safety rates across testing environments with 3, 8, 12, and 16 agents. Our method consistently appears in the bottom-right region, indicating **superior safety-performance co-optimization relative to all baselines**. Its degradation in performance and safety with increasing agent count is minimal, **demonstrating better scalability compared to the baselines**. 198

9.3 Snapshots of multi-agent navigation trajectories at different times using MAD-PINN and baselines. Agents are represented as circles with radius R, indicating the minimum safe distance they must maintain from each other. Smaller dots mark their respective goals. MAD-PINN trajectories show that agents proactively **maintain long-horizon safety** by adjusting their paths to avoid close encounters, rather than enforcing safety reactively (SafeMARL), which could lead to suboptimal behaviours. 200

9.4 **Heatmap of the learned value function with respect to the ego agent's position coordinates.** The other two agents are at $[-0.3, 0.4]$ and $[-0.3, -0.4]$, both moving with velocity $[-1, 0]$, while the ego agent moves with velocity $[1, 0]$ toward its goal at $[-0.5, 0]$. 201

10.1 Presents the architecture of the Safety-driven Latent Dynamics (SaLaD) model. **Training the SaLaD model** the visuomotor observation $\mathcal{O}_i$ is encoded by E_θ into a latent representation z_i. Then, SaLaD predicts the next latent states z_{i+1} as well as the Barrier Certificate over the safe and unsafe datasets for each latent state, and we optimize SaLaD using Equation (10.7). Next, visuomotor observation $\mathcal{O}_{i+1}$ is encoded using target net E_{θ^-} and used as latent targets only during training. . . 213

10.2 This figure demonstrates the results of our framework applied to OpenAI Gym's Pendulum environment. **Fig. 10.2a** illustrates the system diagram, where the light green area represents the safe region (used for sampling safe states, $\mathcal{S}$), and the light red area denotes the unsafe region (used for sampling unsafe states, $\mathcal{U}$). **Fig. 10.2b** shows the representation of safe, unsafe and buffer states. The figure shows that the safe and unsafe regions are well segregated. **Fig. 10.2c** visualizes the safe, unsafe, and buffer points in the $\Theta - \dot{\Theta}$ plane. We observe that the boundary of the trained barrier function (black dots) successfully separates the unsafe and safe regions. **Fig. 10.2d** shows the pendulum's trajectory generated by the learned policy π_θ, where the size of the arrows indicates the magnitude of the control actions produced by the policy. 218

10.3 Presents the results on the velocity-driven quadruped model on PyBullet. Figure 10.3b depicts the x–y plane, where the light green and red regions indicate safe ($\mathcal{S}$) and unsafe ($\mathcal{U}$) states used for sampling. Trajectories starting in the safe set follow the policy π_θ synthesized by our framework. 220

10.4 Empirical safety comparison across methods. The verified SaLaD variant achieves the strongest observed safety performance in the tested settings, while GCIL and PPO-Lagrangian do not enforce the latent CBC condition. The unverified ablation illustrates the empirical benefit of integrating the proposed certificate during training. 221

C.1 This figure shows the α-ϵ plots for different numbers of verification samples, N, and different values of β. 243

C.2 A visual representation of the aerial vehicle 244

C.3 Implementation of the CP-NCBF-QP controller on a Crazyflie drone in Pybullet. The red line denotes the desired reference path, grey lines indicate the safety limits at $x_1 = -1$ and $x_1 = 1$, and the blue line represents the safe trajectory generated by our controller. 246

C.4 Visualization of learned NCBFs for the quadruped navigation at $\phi, x_{o1}, x_{o2}, v_{o1}, v_{o2} = 0$, $r = 0.5$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (with 99.5% safety guarantee), and (Right) CP-NCBF (with 99.9% safety guarantee). Bold black lines denote the zero-level sets. 247

C.5 Snapshots of quadruped navigation in Pybullet with a moving obstacle. The CP-NCBF-QP controller successfully prevents collisions while the agent progresses along its straight-line path. 247

C.6 Dependence of conformal safety scores on the size of the training dataset for (a) ground-vehicle collision avoidance and (b) aerial-vehicle geo-fencing tasks. 250

D.1 **τ-Expectiles:** Figure shows a depiction of the learned Barrier Function for different τ values with Expectile Regression. With increasing value of τ (> 0.5), the feasible region increases, indicating a more accurate representation of the true barrier function. 254

D.2 **Safe Set Volume:** Figure shows a depiction of feasible region (i.e., anything outside B=0 enclosure) for V-OCBF for 3 different heading angles. Note that calculation of the actual Safe Set volume is a cumulative of feasible region over countless such frames along the continuous heading angle (ϕ) dimension. 256

D.3 **Walker-2D.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep. 258

D.4 **Half-Cheetah.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep. 258

D.5 **Ant.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep. 258

E.1 Heatmap of the value function for the ground truth (left) and our method (right). The yellow region represents the unsafe area. Our method successfully captures most of the safe set, indicating that it is not overly conservative while completely recovering the unsafe regions. 276

E.2 This figure presents a comparative analysis of the relationships between ϵ–δ, α–δ, and empirical safety violation rate–δ. As observed, the empirical safety violation consistently remains below the theoretical bound, thereby supporting our theoretical guarantees. As ϵ decreases, the corresponding δ approaches zero, indicating that the learned value function incurs negligible safety violations. 277

E.3 Snapshots of multi-agent navigation trajectories at different time instances using **MPPI**. The trajectories indicate that the agents adopt a **highly conservative strategy** to prevent collisions. Consequently, this leads to a **reduction in performance**, as the agents **end up very far from their respective goals**. 279

E.4 Snapshots of multi-agent navigation trajectories at different time instances using **MPPI-NCBF**. The observed trajectories demonstrate **suboptimal behavior similar to that of the MPPI policy**. Consequently, this results in high-performance costs, indicating its **inability to effectively co-optimize safety and performance.** 280

E.5 Snapshots of multi-agent navigation trajectories at different time instances using SAC-Lag. The trajectories indicate that agents **demonstrate less conservative behavior compared to MPPI and MPPI-NCBF, but they lead to collisions**. These **safety violations are critical** and cannot be disregarded, further **highlighting the limitations of the baseline methods in simultaneously optimizing safety and performance.** 280

List of Tables

5.1 Sensitivity of the smooth unicycle SNCBF to the diffusion magnitude. The $\sigma = 0$ entry reproduces the deterministic Chapter 3 result. 106

5.2 ReLU SNCBF Synthesis Comparison 107

5.3 Comparison of Baseline, Smooth and ReLU SNCBF Synthesis. The smooth SNCBF is the proposed verifiable synthesis in Algorithm 4. The ReLU SNCBF is synthesized by VITL with the efficient verifier proposed in Algorithm 6. 107

5.4 Sensitivity analysis of sampling budget N (and implicitly $\bar{\epsilon}$) on the unicycle system. 108

6.1 Comparison between the HOCBF (6.25) and the proposed C3BF (6.8) for different vehicle models. 125

6.2 Extreme unicycle comparison over 100 cases. Intervention energy is $E_{\text{int}} = \sum_k \|u_k - u_{\text{ref},k}\|_2^2 \Delta t$, which measures the cumulative squared control correction applied by the safety filter. 127

7.1 AGV Collision Avoidance Experiment: Percentage Safe Episodes, Mean Episode Reward and Safe Set Volume (refer Appendix D.2.1) across different methods. Evaluated over 500 episodes and 5 seed values. 150

7.2 Use of Learned Dynamics at the time of Training v/s Inference (Mean over 500 episode rollouts). 152

8.1 Safety rate (%) and mean task cost for the boat navigation case study under varying disturbance levels. A notable trend is that the safety rate remains at 100% when the evaluation disturbance is equal to or smaller than the disturbance used during training, but decreases when the evaluation disturbance exceeds the training disturbance. Additionally, increasing the training disturbance does not lead to a substantial increase in the mean task cost, suggesting that the proposed robust training framework improves safety without significantly degrading task performance. 183

8.2 Safety rate (%) with and without conformal prediction (CP) correction. Each row reports the safety rate evaluated at its respective training disturbance level. 184

8.3 Safety rate (%) and mean task cost for the evader tracking case study with and without evader model uncertainty. A notable trend is that the safety rate remains high when the evaluation disturbance is equal to or smaller than the training disturbance, but decreases when the evaluation disturbance exceeds it. Additionally, increasing the training disturbance does not lead to a substantial increase in the mean task cost, suggesting that the proposed robust training framework, when trained with high disturbance values, improves safety without significantly degrading task performance. . 186

8.4 Safety rate (%) with and without conformal prediction (CP) correction for the pursuer-evader tracking case study. Each row reports the safety rate evaluated at its respective training disturbance level. 186

9.1 Average safety rates and safe-scenario percentages across varying numbers of agents. Our method maintains consistently high values, indicating collision-free execution of every agent for nearly all initial configurations and demonstrating **effective safety-performance co-optimization at the agent and system level**. 200

9.2 Effect of increasing agent count and environment size. Our method **co-optimizes safety and performance even in large multi-agent environments**, despite training on smaller environments with fewer agents. 202

9.3 Ablation study on the impact of the proposed neighbor selection strategy. By effectively **capturing safety-critical inter-agent interactions among neighbors**, it enables improved safety–performance co-optimization. 204

11.1 Summary of the guarantee type, certified or evaluated statement, and key assumptions for each technical chapter. The rows distinguish deterministic invariance, distribution-free probabilistic coverage, conditional learned-component assurances, and empirical evaluation. 225

C.1 Hyperparameters for the proposed algorithm 248

D.1 Threshold velocity $V_{threshold}$, time step size Δt and action space $\mathbb{U}$ for each MuJoCo based Safety Gymnasium [77] environment as per official documentation. 257

D.2 Safety Rate (%) reported for various model dimensions. Columns represent Width: neurons in each hidden layer, and Rows represent number of layers. 259

D.3 Safety Rate (%) reported for Robustness Testing of **learned Dynamics** for Autonomous Ground Vehicle by adding upto 5%, 10% and 20% of the possible control action as noise. 260

D.4 Safety Rate (%) reported for Autonomous Ground Vehicle by varying τ in the expectile regression. 260

D.5 Reward and Cost across different τ expectile values for selected MuJoCo environments 261

D.6 Effect of Dynamics Model Accuracy on Safety Rates at the time of Inference (Mean over 500 episode rollouts). 262

D.7 Boat Navigation Experiment: Percentage Safe Episodes against additional baseline methods. Evaluated over 500 episodes and 5 seed values. 263

D.8 Hyperparameters for the Proposed (V-OCBF) Network, and Learned Dynamics Model. 265

D.9 Hyperparameters for all the Baselines network. Optimizer: Adam, LR: 1×10^{-3}, Activation: ReLU (applies to all rows). For COptiDICE and BEAR-Lag refer OSRL, [100], for the official implementations. For FISOR refer [204] for official implementation. 266

E.1 Hyperparameters for the proposed algorithm 281

E.2 Hyperparameters for MPPI Baselines 282

E.3 General Hyperparameters of SAC in our experiments 282

E.4 General Hyperparameters of PPO in our experiments 283

E.5 CPO Hyperparameters from OmniSafe Configuration used for our experiments . . . 284

Glossary

Abbreviations

CBF	**C**ontrol **B**arrier **F**unction
CLF	**C**ontrol **L**yapunov **F**unction
HOCBF	**H**igher-**O**rder **C**ontrol **B**arrier **F**unction
NCBF	**N**eural **C**ontrol **B**arrier **F**unction
SNCBF	**S**tochastic **N**eural **C**ontrol **B**arrier **F**unction
SCBF	**S**tochastic **C**ontrol **B**arrier **F**unction
CBVF	**C**ontrol **B**arrier-**V**alue **F**unction
C3BF	**C**ollision **C**one **C**ontrol **B**arrier **F**unction
VITL	**V**erification-**I**n-**T**he-**L**oop
QP	**Q**uadratic **P**rogram
SDE	**S**tochastic **D**ifferential **E**quation
ODE	**O**rdinary **D**ifferential **E**quation
HJB	**H**amilton–**J**acobi–**B**ellman
HJI	**H**amilton–**J**acobi–**I**saacs
HJR	**H**amilton–**J**acobi **R**eachability
NN	**N**eural **N**etwork
RL	**R**einforcement **L**earning
SOS	**S**um-**O**f-**S**quares
SMT	**S**atisfiability **M**odulo **T**heories
LMI	**L**inear **M**atrix **I**nequality
ISS	**I**nput-to-**S**tate **S**tability
ISSf	**I**nput-to-**S**tate **S**afety
PINN	**P**hysics-**I**nformed **N**eural **N**etwork
PDE	**P**artial **D**ifferential **E**quation
ReLU	**Re**ctified **L**inear **U**nit
UGV	**U**nmanned **G**round **V**ehicle

Symbols

Sets

$\mathbb{R}^n$	Real n-dimensional Euclidean space	
$\mathcal{X}$	State space / domain	$\subseteq \mathbb{R}^n$
$\mathcal{C}$	Safe set (super-0-level set of B)	$\{x : B(x) \geq 0\}$
$\mathcal{X}_S$	Initial / known safe set	$\subseteq \mathcal{C}$
$\mathcal{X}_U$	Unsafe set	$\subseteq \mathcal{X} \setminus \mathcal{C}$
$\mathcal{U}$	Admissible control input set	$\subseteq \mathbb{R}^m$
$\mathcal{D}$	Super-0-level set of B / data set	
$\partial\mathcal{C}$	Boundary of safe set	$\{x : B(x) = 0\}$
$\mathrm{Int}(\mathcal{C})$	Interior of safe set	$\{x : B(x) > 0\}$
$\mathcal{D}(\mathcal{A})$	Domain of infinitesimal generator	

State and Control Variables

$x \in \mathbb{R}^n$	System state
$u \in \mathbb{R}^m$	Control input
t	Continuous time
x_0	Initial state
$u_{\mathrm{ref}}(x)$	Reference / nominal controller
$u^*(x)$	Optimal safe controller (CBF-QP solution)

System Dynamics

$f(x)$	Drift vector field	$\mathbb{R}^n \to \mathbb{R}^n$
$g(x)$	Control input matrix	$\mathbb{R}^n \to \mathbb{R}^{n \times m}$
$\sigma(x)$	Diffusion matrix (stochastic systems)	$\mathbb{R}^n \to \mathbb{R}^{n \times p}$
W_t	Standard Brownian motion	$\in \mathbb{R}^p$

Barrier and Lyapunov Functions

$B(x)$	Control Barrier Function (CBF)	$\mathbb{R}^n \to \mathbb{R}$
$\tilde{B}(x \mid \theta)$	Neural CBF (NCBF)	parameterised by θ
$V(x)$	Lyapunov / Value function	$\mathbb{R}^n \to \mathbb{R}$
$h(x)$	Safety constraint function	defines $\mathcal{X}_S$

Comparison Functions

$\alpha(\cdot)$	Extended class-$\mathcal{K}_\infty$ function	CBF decay rate
$\kappa(\cdot)$	Class-$\mathcal{K}$ function	
$\beta(\cdot, \cdot)$	Class-$\mathcal{KL}$ function	

Differential Operators

$\nabla B(x)$	Gradient of B

$\nabla^2 B(x)$	Hessian of B	
$L_f B(x)$	Lie derivative of B along f	$= \nabla B^\top f$
$L_g B(x)$	Lie derivative of B along g	$= \nabla B^\top g$
$\mathcal{A}B(x)$	Infinitesimal generator	(stochastic systems)
$\mathrm{tr}(\cdot)$	Matrix trace	
$\mathrm{diag}(\cdot)$	Diagonal matrix operator	

Neural Network

θ	Trainable parameters (weights and biases)
L	Number of network layers
M_i	Number of neurons in layer i
$\mathbf{W}, \mathbf{r}$	Weight matrix and bias vector
$\sigma(\cdot)$ (activation)	Activation function (ReLU, Softplus, etc.)
L_B	Lipschitz bound on $\tilde{B}$
L_{dB}	Lipschitz bound on $\nabla \tilde{B}$
$L_{\max}$	Maximum Lipschitz constant

Optimisation and Training

ψ	Safety margin (decision variable)
ψ^*	Optimal safety margin
$\bar{\epsilon}$	Maximum sampling distance
N	Number of data samples
$\mathcal{L}_\theta$	Total training loss
Λ	Diagonal matrix in LMI constraint
$M(\theta, \Lambda)$	LMI certificate matrix
$\mathbb{1}_{\mathcal{C}}$	Indicator function

Chapter 1

Introduction

Autonomous systems are steadily transitioning from structured laboratory environments into open-world deployment. Applications such as autonomous driving [24], aerial robotics [163], assistive manipulation, and multi-agent coordination [161] increasingly require operation in dynamic, uncertain, and human-populated environments. In such settings, reliability is not optional: large-scale deployment demands systems that are not only capable but trustworthy.

Trustworthy autonomy rests on two fundamental pillars: safety and performance. Safety requires that catastrophic states are provably avoided [8]. Performance requires that task objectives are achieved efficiently and robustly. Systems that are safe but overly conservative are impractical, while systems that are performant but unsafe are unacceptable. Real-world autonomy demands both simultaneously [59].

The idea that autonomous machines must earn human trust through explicit behavioral guarantees is not new. In his *Handbook of Robotics*, Isaac Asimov proposed the Three Laws of Robotics[1], which require that a robot must not harm humans, must obey legitimate commands, and must preserve its own existence, in that order of priority. More than safety rules, these laws articulate conditions under which humans can *trust* a machine: the robot itself is responsible for ensuring that its actions satisfy verifiable constraints in real time.

Asimov's formulation, however, implicitly assumes that robots possess the reasoning capabilities required to obey such laws under all circumstances. In practice, achieving this level of trustworthiness is profoundly difficult. Modern autonomous systems operate under model uncertainty, partial observability, stochastic disturbances, and high-dimensional sensory inputs [90]. Building a system that humans can justifiably trust, one whose control policy satisfies safety constraints under such uncertainty

[1] I. Asimov, *I, Robot*, Gnome Press, 1950.

without sacrificing performance, remains a central challenge in control and machine learning [44, 24].

Historically, trust in autonomous systems has been established through conservative hardware design, restricted operating envelopes, or continuous human supervision. While effective, these mechanisms limit autonomy and reduce system capability. A more scalable path to trustworthy autonomy is *algorithmic safety*: embedding verifiable safety constraints directly into the control architecture so that the system enforces them during real-time decision-making, providing trust through formal guarantees rather than operational restrictions [8, 24].

This thesis investigates how such algorithmic guarantees can underpin trustworthy autonomy in uncertain environments. Specifically, it asks: *How can we learn control policies that are provably safe yet scale to complex, uncertain systems?* The following section examines why existing approaches, from both control theory and machine learning, fall short of this goal when considered in isolation.

1.1 Formal Safety and Scalable Learning

Safety-critical control has historically been addressed through analytical methods grounded in dynamical systems theory. Lyapunov functions provide stability guarantees for nonlinear systems [82], Control Barrier Functions (CBFs) enforce forward invariance of safe sets through inequality constraints on the control input [8, 6], and Hamilton–Jacobi (HJ) reachability analysis computes maximal safe sets under worst-case disturbances [12, 108]. If the system state begins in a certified safe set, these tools guarantee that it remains safe for all future time. Higher-order extensions [187, 112], stochastic variants [37, 121, 73], and robust formulations [75, 191] have further expanded the applicability of barrier-based safety.

However, classical safety synthesis presumes accurate knowledge of system dynamics, explicit specification of safety constraints, and tractable state-space dimensionality. As autonomy moves toward high-dimensional, perception-driven systems operating under model uncertainty, these assumptions become restrictive. Constructing barrier functions manually is labor-intensive and model-specific [44]. HJ reachability scales poorly with state dimension due to the curse of dimensionality in grid-based PDE solvers [43, 13]. Analytical verification becomes intractable when safety must depend on learned representations [47].

In contrast, learning-based methods provide flexibility and scalability. Reinforcement learning [137, 64], representation learning [84, 65], and learned control policies [90, 80] enable systems to operate under unknown dynamics and learn value

functions from data [109, 182]. Yet these approaches typically lack worst-case safety guarantees [24, 59, 202], rely on asymptotic performance claims, and may generate unsafe behaviors during training or deployment [130].

The resulting tension is fundamental: *classical methods guarantee safety but not scalability, while learning-based methods achieve scalability but not safety* [24].

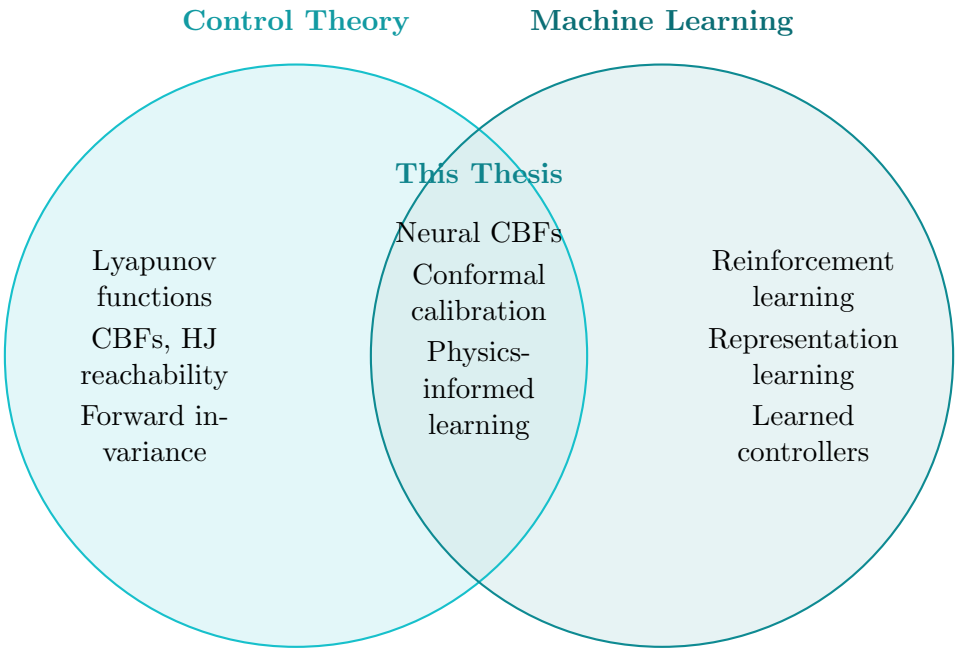


Figure 1.1: Classical control theory (left) provides rigorous safety guarantees but requires accurate models and scales poorly; machine learning (right) offers flexibility and scalability without formal assurances. This thesis develops methods at the intersection, combining learning-based safety and optimality tools with formal guarantees.

This thesis addresses the question:

How can we combine the formal safety guarantees of classical control with the scalability of learning-based methods?

1.2 From Analytical to Learned Safety

The key conceptual shift underlying this work is the move from hand-crafted to learned safety. Classically, safe control relies on constructing functions over state that enforce safety or certify optimality: barrier functions [8, 121] guarantee constraint satisfaction, and value functions [12, 56] certify optimality or viability. These are traditionally designed analytically, requiring known dynamics and domain expertise. Recent work has begun to bridge this analytical tradition with learning-based methods by parameterizing barrier and value functions as parametric functions [72, 111] or as neural

networks [44, 79, 132, 118], training them from sampled trajectories [123, 144], and applying post-hoc verification [38, 198].

This thesis advances this line of research by demonstrating that neural barrier and value functions can be not only trained from finite data but also verified or calibrated *during or after training* to yield control policies with formal safety guarantees. Specifically, verification is achieved through three complementary paradigms: Lipschitz-based global certification [165], split conformal prediction [158, 10], and verification-in-the-loop synthesis [197].

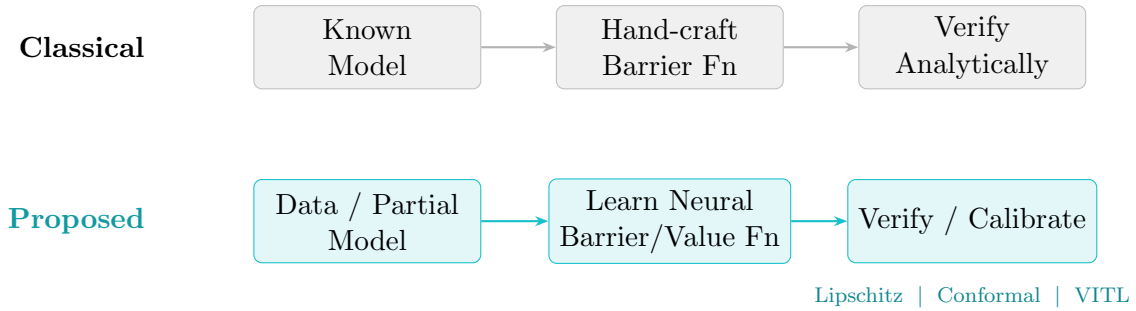


Figure 1.2: From analytical to learned safety. Classical safety synthesis (top, grey) requires known dynamics and hand-crafted barrier functions verified analytically. This thesis (bottom, teal) learns neural barrier and value functions from data and verifies or calibrates them via Lipschitz bounds [165], conformal prediction [158], or verification-in-the-loop synthesis [197].

Under this perspective, trustworthy autonomy becomes the problem of learning safe policies under uncertainty [44].

1.3 Central Claim

Across deterministic, stochastic, offline, optimal, multi-agent, and perception-driven settings, this thesis supports the following claim:

Trustworthy control policies for nonlinear systems can be learned from finite data, using neural barrier and value functions that are formally verified or calibrated to retain rigorous safety guarantees under uncertainty.

This reframes safe autonomy not as a conflict between learning and control, but as a synthesis problem: learning trustworthy policies whose safety properties are formally guaranteed. The resulting framework provides scalable safety synthesis for high-dimensional systems, finite-sample reliability under distributional uncertainty

via conformal prediction [10, 175], compatibility with offline data regimes [91, 87], real-time deployment feasibility, and extension to perception-based control [45, 173]. By integrating control-theoretic rigor with statistical learning principles, this work lays a foundation for trustworthy autonomy in complex, uncertain environments.

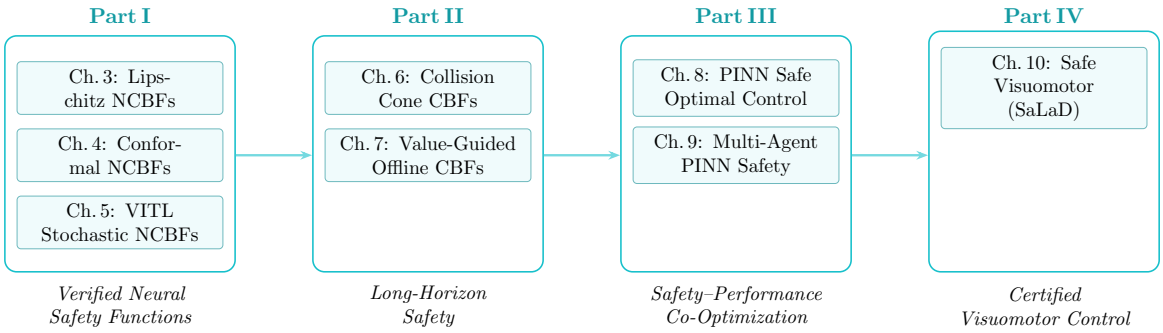


Figure 1.3: Thesis architecture. The dissertation progresses through four parts: from learning and verifying neural safety functions (Part I), through long-horizon and data-driven safety (Part II), to joint safety–performance optimization (Part III), and finally certified visuomotor control from raw perception (Part IV). Arrows indicate the conceptual progression.

1.4 Thesis Overview

The preceding sections have identified a fundamental gap: classical control theory provides rigorous safety guarantees but does not scale, while learning-based methods scale but lack formal safety assurances. This thesis bridges that gap by developing methods that learn trustworthy control policies with provable safety guarantees, without sacrificing scalability or performance. The remainder of this chapter summarizes each contribution and its place in the overall arc. This dissertation is organized into eleven chapters grouped into four thematic parts, preceded by a background chapter that develops the shared mathematical foundations.

Chapter 2 presents the theoretical background required for the remainder of the thesis, including control-affine dynamical systems, Lie derivatives, forward invariance, Control Barrier Functions (including higher-order [187], stochastic [37], and robust [75] variants), Hamilton–Jacobi Reachability [12], Physics-Informed Neural Networks [127], Reinforcement Learning (online and offline) [91, 64], conformal prediction [10], and latent-space representations for visuomotor control [65].

Part I: On Learning Verified Neural CBFs

Constructing Control Barrier Functions by hand is labour-intensive, model-specific, and does not scale to complex or high-dimensional safety specifications. This part develops learning-based approaches that overcome these limitations through three complementary verification paradigms.

Chapter 3 presents an algorithm for synthesizing formally verified continuous-time Neural Control Barrier Functions for control-affine systems in a single step [165]. A sample-based learning framework ensures efficacy across the entire state space from only a finite number of data points by enforcing Lipschitz bounds on the neural network and its Jacobian *during* training, eliminating the need for post-hoc verification. While powerful, the Lipschitz-based guarantees introduce conservatism through dependence on global bounds and are restricted to single-hidden-layer architectures.

To address this, Chapter 4 develops *Conformal-Prediction NCBFs (CP-NCBFs)* [158], which replace Lipschitz-based verification with split conformal prediction [10, 175], yielding finite-sample, distribution-free safety guarantees that are significantly less conservative and scale effectively with neural network size. An iterative training procedure uses the conformal score to refine the safety margin, converging when the barrier satisfies its conditions at the desired probability level.

Chapter 5 extends the synthesis of Neural CBFs to *stochastic* control-affine systems [165, 197]. It considers two cases: smooth SNCBFs with twice-differentiable activation functions, and SNCBFs that use ReLU activations. For smooth SNCBFs a verification-free synthesis framework based on Lipschitz bounds is developed; for both smooth and ReLU SNCBFs a *Verification-In-The-Loop* (VITL) framework replaces dense sampling with NLP-based verifiers that identify precisely where the stochastic barrier conditions fail, enabling deeper architectures without LMI constraints. Together, these three chapters establish a complete pipeline for learning and verifying neural safety functions, but enforce safety in a single-step, myopic sense.

Part II: Safety Beyond One Step

Standard CBF conditions enforce safety in a myopic, single-time-step sense and can therefore be overly conservative or infeasible when control authority is limited or future hazards are foreseeable. This part develops methods that embed temporal look-ahead into the safety framework.

Chapter 6 presents *Collision Cone Control Barrier Functions (C3BFs)* [153, 154], which combine geometric collision cones [26, 27] with CBFs to provide velocity-level

look-ahead for collision avoidance. C3BF formulations are developed for unicycle, bicycle, and quadrotor platforms [163, 153], and the framework is extended to cluttered environments via Polygonal Cone Control Barrier Functions (PolyC2BFs) [155]. However, the collision cone construction is specific to collision avoidance and requires manual, model-dependent design, motivating a more general approach.

Chapter 7 introduces *Value-Guided Offline Control Barrier Functions (V-OCBF)* [168], which learns a Neural CBF from offline data in a model-free setting that accounts for future safety violations over an infinite horizon. The key idea is to use HJ-reachability-style Bellman updates [12, 56] to propagate long-horizon safety information into the barrier function, enabling certified safety filtering for arbitrary safety specifications without requiring known dynamics or online rollouts.

Part III: Bridging Safety with Performance

The methods developed in Parts I and II treat safety and task performance as separate objectives: the barrier function ensures constraint satisfaction, while a nominal controller pursues the task. This decoupled design can lead to overly conservative behavior or suboptimal performance. Part III addresses this fundamental tension by formulating the joint safety–performance objective as a *state-constrained optimal control problem* and solving it using physics-informed machine learning.

Chapter 8 reformulates the state-constrained optimal control problem via an epigraph transformation [142] and solves the resulting Hamilton–Jacobi–Bellman PDE with physics-informed neural networks [127, 160], yielding value functions and policies that are simultaneously safe and near-optimal. Conformal prediction provides formal safety guarantees and performance bounds [10].

Chapter 9 extends this framework to cooperative multi-agent settings [161], introducing a decentralized formulation with fixed observation size and a reachability-inspired neighbor-selection strategy that identifies the most safety-critical interactions.

Part IV: Safety in Latent Spaces

The final part of this dissertation addresses safe decision-making directly from high-dimensional observations such as images. Chapter 10 introduces the *Safety-based Latent Dynamics (SaLaD)* model [159], which learns a compact latent representation from raw visual inputs while preserving safety-relevant structure [65, 84]. A Neural

Control Barrier Function is synthesized in this latent space alongside a safe policy via semi-supervised learning, with inherent completeness guarantees that eliminate the need for post-hoc verification [45]. A consistency condition for non-zero dynamics ensures the safe policy satisfies the barrier decrease condition at every time step.

The dissertation concludes with Chapter 11, which synthesizes the contributions as a narrative tracing the thesis arc, identifies two unifying methodological threads (conformal prediction and physics-informed learning) that bind the chapters into a coherent framework, and outlines five research directions for future work.

Chapter 2

Background and Preliminaries

Autonomous systems operating in uncertain, dynamic, and partially unknown environments require a mathematical framework that integrates control-theoretic guarantees with learning-enabled adaptability. This chapter develops the foundational concepts used throughout the dissertation. The exposition begins with deterministic and stochastic dynamical systems and their stability properties, continues with the theory of invariance and safety, introduces Control Barrier Functions (CBFs), in their nominal, stochastic, and robust forms, as a fundamental tool for guaranteeing forward invariance, and then develops the analytical structures on which learning-based safety certificates are later built. The second half of the chapter covers Hamilton–Jacobi reachability as a complementary framework for worst-case safety verification, the unification of CBFs and reachability through Control Barrier-Value Functions, and the reinforcement learning and statistical tools (conformal prediction, world models) that enable data-driven instantiation of these safety concepts. Particular emphasis is placed on the interaction between smooth dynamical structure, optimization-based controllers, and differentiable safety certificates, since these themes recur in subsequent chapters.

2.1 Dynamical Systems

We consider a system evolving on a state space $\mathbb{R}^n$ with control inputs in $\mathbb{R}^m$. The majority of robotic and mechanical systems studied in this thesis admit a *control-affine* representation.

Definition 2.1.1 (Control-Affine System)**.** A continuous-time dynamical system

is said to be *control-affine* if its dynamics admit the form

$$\dot{x} = f(x) + g(x)u, \tag{2.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and the functions $f : \mathbb{R}^n \to \mathbb{R}^n$ (drift) and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ (input matrix) are locally Lipschitz continuous.

Under local Lipschitz continuity, the Cauchy–Lipschitz (Picard–Lindelöf) theorem ensures existence and uniqueness of solutions for any measurable, locally bounded control $u(t)$. Control-affine systems encompass robotic manipulators, ground vehicles, quadrotor models, and simplified representations of legged locomotion. The affine dependence on u is particularly advantageous because it renders the CBF safety constraint (Section 2.3) linear in the control input, enabling efficient quadratic-program-based synthesis.

Discrete-time systems appear frequently in reinforcement learning and numerical control. Such systems evolve according to

$$x_{k+1} = F(x_k, u_k), \tag{2.2}$$

or stochastically,

$$x_{k+1} \sim P(\cdot \mid x_k, u_k). \tag{2.3}$$

The continuous-time and discrete-time formulations share many conceptual parallels, but continuous dynamics permit more refined notions of differentiability, barriers, and value functions, which will be central in later chapters.

2.1.1 Comparison Functions

Many stability and safety conditions are stated in terms of comparison functions. We collect the standard definitions here for reference.

Definition 2.1.2 (Class $\mathcal{K}$, $\mathcal{K}_\infty$, and $\mathcal{KL}$)**.**

1. A continuous function $\alpha : [0, \infty) \to [0, \infty)$ is of *class* $\mathcal{K}$ if $\alpha(0) = 0$ and α is strictly increasing.
2. A class-$\mathcal{K}$ function is of *class* $\mathcal{K}_\infty$ if, in addition, $\alpha(r) \to \infty$ as $r \to \infty$.
3. An *extended class-*$\mathcal{K}_\infty$ function is defined on all of $\mathbb{R}$ (i.e., $\alpha : \mathbb{R} \to \mathbb{R}$), is strictly increasing, satisfies $\alpha(0) = 0$, and $\alpha(r) \to \pm\infty$ as $r \to \pm\infty$.
4. A continuous function $\beta : [0, \infty) \times [0, \infty) \to [0, \infty)$ is of *class* $\mathcal{KL}$ if $\beta(\cdot, t)$ is class $\mathcal{K}$ for each fixed t, and $\beta(r, \cdot)$ is decreasing with $\beta(r, t) \to 0$ as $t \to \infty$

for each fixed $r > 0$.

Extended class-$\mathcal{K}_\infty$ functions appear in the CBF condition (Definition 2.3.1), where they control the rate at which the barrier function is allowed to decrease. Class-$\mathcal{KL}$ functions characterise the transient behaviour in stability and Input-to-State Stability (ISS).

2.1.2 Lie Derivatives and Relative Degree

For a differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, the *Lie derivatives* along f and g are defined as

$$L_f h(x) = \nabla h(x)^\top f(x), \qquad L_g h(x) = \nabla h(x)^\top g(x). \tag{2.4}$$

These derivatives describe how $h(x)$ evolves under the system dynamics. Higher-order Lie derivatives are obtained by iterating:

$$L_f^k h(x) = L_f(L_f^{k-1} h)(x), \qquad L_f^0 h = h. \tag{2.5}$$

Definition 2.1.3 (Relative Degree)**.** The function h has *relative degree* r with respect to (2.1) if

$$L_g L_f^k h(x) = 0 \quad \text{for all } x \text{ and } k = 0, 1, \ldots, r-2, \qquad L_g L_f^{r-1} h(x) \neq 0. \tag{2.6}$$

When the relative degree $r = 1$, the control input u directly affects $\dot{h} = L_f h + L_g h\, u$, and the standard CBF condition applies immediately. When $r > 1$, the control does not appear until the r-th time derivative of h, and one must employ higher-order CBF constructions (Section 2.3.1).

2.1.3 Stability in the Sense of Lyapunov

Stability theory provides the classical framework for analysing the long-term behaviour of dynamical systems and is tightly connected to the barrier function methodology developed later.

Definition 2.1.4 (Equilibrium Stability)**.** An equilibrium $x^* = 0$ of $\dot{x} = f(x)$ is

1. *stable* if for every $\varepsilon > 0$ there exists $\delta > 0$ such that $\|x(0)\| < \delta$ implies $\|x(t)\| < \varepsilon$ for all $t \geq 0$;
2. *asymptotically stable* if it is stable and there exists $\delta > 0$ such that $\|x(0)\| <$

δ implies $\|x(t)\| \to 0$ as $t \to \infty$.

Theorem 2.1.5 (Lyapunov's Direct Method [82])**.** *Consider the system $\dot{x} = f(x)$ with $f(0) = 0$, and let $V : \mathcal{D} \to \mathbb{R}$ be a continuously differentiable function defined on a domain $\mathcal{D} \ni 0$ satisfying*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|), \qquad \dot{V}(x) = \nabla V(x)^\top f(x) \leq -\alpha_3(\|x\|), \tag{2.7}$$

where α_1, α_2 are class-$\mathcal{K}$ and α_3 is continuous and positive definite. Then the origin is asymptotically stable. If $\mathcal{D} = \mathbb{R}^n$ and $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$, the stability is global.

This is a classical result; a proof may be found in [82].

Remark 2.1.6 (Lyapunov Functions vs. Barrier Functions). *Strong* Lyapunov functions—functions whose derivative along trajectories is negative definite, as in Theorem 2.1.5—certify *asymptotic convergence* to an equilibrium; a function that merely decreases along trajectories certifies stability in the sense of Lyapunov but not necessarily convergence. Barrier functions, in contrast, certify *avoidance* of an unsafe set (the function remains non-negative). Despite this dual perspective, the two share a common mathematical structure: both reduce properties of trajectories (stability or safety) to pointwise derivative conditions. Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) can be combined in a unified QP to simultaneously achieve stability and safety [8].

2.2 Safety and Invariance

Safety is formalized by ensuring that trajectories remain in a desirable set for all time. Let $B : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable, and define the set

$$\mathcal{C} = \{x \in \mathbb{R}^n : B(x) \geq 0\}. \tag{2.8}$$

Definition 2.2.1. A set $\mathcal{C}$ is *forward invariant* under a control law $u = \pi(x)$ if every trajectory starting in $\mathcal{C}$ remains in $\mathcal{C}$ for all future time.

Forward invariance is tightly connected to the geometric evolution of $B(x(t))$.

Theorem 2.2.2 (Nagumo [19]). *Let $\mathcal{C}$ be the safe set defined in (2.8). Suppose f is continuous and B is continuously differentiable with $\nabla B(x) \neq 0$ for all $x \in \partial\mathcal{C}$. Then $\mathcal{C}$ is forward invariant under $\dot{x} = f(x)$ if and only if, for all $x \in \partial\mathcal{C}$, the vector field satisfies*

$$L_f B(x) \geq 0. \tag{2.9}$$

This classical result originates with Nagumo; see [19] for a modern treatment of set-invariance theory and [6] for its role in the CBF literature.

Nagumo's theorem establishes a geometric invariance condition, but it does not account for control inputs. Control Barrier Functions extend this idea to controlled systems.

2.3 Control Barrier Functions

Definition 2.3.1. A continuously differentiable function $B : \mathbb{R}^n \to \mathbb{R}$ is a *Control Barrier Function* (CBF) for the system (2.1) if there exists an extended class-$\mathcal{K}$ function α such that, for all $x \in \mathbb{R}^n$,

$$\sup_{u \in \mathbb{R}^m} \left(L_f B(x) + L_g B(x) u + \alpha(B(x)) \right) \geq 0. \tag{2.10}$$

Condition (2.10) ensures that it is always possible to choose a control u that prevents $B(x(t))$ from decreasing in a way that would violate safety. Given a nominal control $u_{\text{ref}}(x)$, safety is enforced by solving the quadratic program

$$u^\star(x) = \arg\min_{u \in \mathbb{R}^m} \|u - u_{\text{ref}}(x)\|^2 \quad \text{subject to} \quad L_f B(x) + L_g B(x) u + \alpha(B(x)) \geq 0. \tag{2.11}$$

Proposition 2.3.2. *If B is a CBF and $u^\star(x)$ solves (2.11), then $\mathcal{C}$ is forward invariant.*

Proof. Since B is a CBF, the constraint set of (2.11) is non-empty for every $x \in \mathbb{R}^n$ by Definition 2.3.1. Therefore, a solution $u^\star(x)$ exists and satisfies

$$L_f B(x) + L_g B(x) u^\star(x) + \alpha(B(x)) \geq 0 \quad \text{for all } x. \tag{2.12}$$

Along the closed-loop trajectory $\dot{x} = f(x) + g(x)u^\star(x)$, the time derivative of B is

$$\dot{B}(x(t)) = \nabla B(x(t))^\top \big(f(x(t)) + g(x(t))u^\star(x(t))\big) = L_f B(x(t)) + L_g B(x(t))u^\star(x(t)). \tag{2.13}$$

Combining with (2.12), we obtain the differential inequality

$$\dot{B}(x(t)) \geq -\alpha(B(x(t))). \tag{2.14}$$

We now invoke the Comparison Lemma [82].

Lemma 2.1 (Comparison Lemma)**.** *Consider the scalar differential equation $\dot{y} = -\alpha(y)$ with $y(0) = y_0$. If a differentiable function $B(x(t))$ satisfies $\dot{B}(x(t)) \geq -\alpha(B(x(t)))$ and $B(x(0)) = y_0$, then $B(x(t)) \geq y(t)$ for all $t \geq 0$ in the domain of existence.*

Applying Lemma 2.1 with $y_0 = B(x(0))$, the function $B(x(t))$ is bounded below by the solution $y(t)$ of $\dot{y} = -\alpha(y)$ with $y(0) = B(x(0))$.
Since α is an extended class-$\mathcal{K}$ function, we have $\alpha(0) = 0$. Therefore, $y \equiv 0$ is an equilibrium of $\dot{y} = -\alpha(y)$. Moreover, for $y > 0$, $\alpha(y) > 0$ implies $\dot{y} < 0$, so solutions starting at $y_0 > 0$ decrease toward zero but never cross it. For $y_0 = 0$, the solution remains at zero.
Formally, suppose $x(0) \in \mathcal{C}$, i.e., $B(x(0)) \geq 0$. Then $y_0 \geq 0$, and the comparison solution satisfies $y(t) \geq 0$ for all $t \geq 0$ (since $y = 0$ is an equilibrium that cannot be crossed from above). By the comparison bound,

$$B(x(t)) \geq y(t) \geq 0 \quad \text{for all } t \geq 0. \tag{2.15}$$

□

Control Barrier Functions unify classical invariance theory with optimization-based control, and they serve as the foundation for the learning-based barrier constructions developed in Chapters 3–7.

2.3.1 Higher-Order Control Barrier Functions

When the barrier function B has relative degree $r > 1$ with respect to (2.1) (Definition 2.1.3), the control input u does not appear in $\dot{B}$, so the standard CBF condition cannot be directly enforced. Higher-Order CBFs (HOCBFs) [187] address this by constructing a sequence of auxiliary functions.

Definition 2.3.3 (Higher-Order CBF [187])**.** Let B have relative degree r and define the sequence of functions

$$\begin{aligned} \psi_0(x) &= B(x), \\ \psi_i(x) &= \dot{\psi}_{i-1}(x) + \alpha_i(\psi_{i-1}(x)), \quad i = 1, \dots, r, \end{aligned} \tag{2.16}$$

where each α_i is a class-$\mathcal{K}_\infty$ function. Then B is a *Higher-Order CBF* of degree r if

$$\sup_{u \in \mathbb{R}^m} \left[L_f^r B(x) + L_g L_f^{r-1} B(x)\, u + O(\psi_0, \dots, \psi_{r-1}) + \alpha_r(\psi_{r-1}(x)) \right] \geq 0 \tag{2.17}$$

for all x in the domain, where $O(\cdot)$ collects the lower-order terms.

The HOCBF condition ensures that *all* intermediate sets $\mathcal{C}_i = \{x : \psi_i(x) \geq 0\}$, $i = 0, \dots, r-1$, are forward invariant, and hence the original safe set $\mathcal{C}_0 = \{x : B(x) \geq 0\}$ is preserved. Systems with position-level constraints but velocity-level actuation (e.g., distance-based collision avoidance for double-integrator models) naturally have relative degree two and require HOCBFs. The Collision Cone formulation in Chapter 6 provides an alternative approach that reduces the effective relative degree by incorporating velocity information directly into the barrier function.

2.3.2 Feasibility of the CBF–QP

A practical concern when deploying CBF-based controllers is that the QP (2.11) may become *infeasible* when control inputs are bounded, i.e., $u \in \mathcal{U}$ for a compact set $\mathcal{U}$.

Remark 2.3.4 (When Does the CBF–QP Fail?). Feasibility of the CBF–QP at a state x requires

$$\max_{u \in \mathcal{U}} \left[L_f B(x) + L_g B(x) u \right] \geq -\alpha(B(x)). \tag{2.18}$$

When B is handcrafted, there is no guarantee that this condition holds for all $x \in \mathcal{C}$—especially near the boundary $\partial\mathcal{C}$ where $B(x) \approx 0$ and $L_f B(x)$ may be strongly negative. Infeasibility signals that the chosen barrier function defines a safe set that is *too large* to be rendered invariant with the available actuation authority. Hamilton–Jacobi reachability (Section 2.6.1) provides the exact maximal set for which a safe controller exists, but its computation is limited to low-dimensional systems.

Two common remedies are: (i) relaxing the CBF constraint with a slack variable [8], which sacrifices the hard safety guarantee but preserves QP feasibility; and

(ii) learning a valid CBF whose zero-superlevel set is an under-approximation of the true viability kernel, as pursued in Chapters 3–7 of this thesis.

2.3.3 Unification with Control Lyapunov Functions

When both stability and safety are desired, the CBF constraint can be combined with a Control Lyapunov Function (CLF) constraint in a single QP [8]:

$$u^\star = \arg\min_{u,\,\delta} \|u - u_{\text{ref}}\|^2 + p\,\delta^2 \quad \text{s.t.} \quad \begin{cases} L_f V + L_g V\, u \le -\gamma(V) + \delta, \\ L_f B + L_g B\, u \ge -\alpha(B), \end{cases} \tag{2.19}$$

where V is a CLF, γ is a class-$\mathcal{K}$ function, and δ is a relaxation variable penalised by $p > 0$. The relaxation is applied only to the CLF (stability) constraint, ensuring that safety always takes priority.

Having developed the CBF framework for nominal deterministic systems, we now extend the analysis to two important classes of uncertainty: stochastic noise (Section 2.4) and bounded deterministic disturbances (Section 2.5).

2.4 Stochastic Systems and Barrier Conditions

Robotic systems often experience noise due to sensing, actuation errors, or environmental disturbances. We model such systems using stochastic differential equations and develop the associated safety machinery in detail.

2.4.1 Stochastic Dynamical Systems

Consider the controlled stochastic dynamical system:

$$dx = (f(x) + g(x)u)\,dt + \sigma(x)dW_t, \tag{2.20}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, f and g are locally Lipschitz, $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times p}$ is the diffusion matrix, and $W_t \in \mathbb{R}^p$ is a standard p-dimensional Brownian motion on a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\ge 0}, \mathbb{P})$.

Under local Lipschitz and linear growth conditions on the coefficients, the SDE admits a unique strong solution that is adapted to the filtration $\{\mathcal{F}_t\}$.

2.4.2 Itô's Lemma

The key tool for analyzing functions of stochastic processes is Itô's lemma, which generalizes the chain rule to stochastic calculus.

Theorem 2.4.1 (Itô's Lemma)**.** *Let $B : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable, and let $x(t)$ satisfy the SDE (2.20). Then,*

$$dB(x) = \left[\nabla B(x)^\top \big(f(x) + g(x)u\big) + \frac{1}{2}\operatorname{tr}\big(\sigma(x)^\top \nabla^2 B(x)\, \sigma(x)\big)\right] dt + \nabla B(x)^\top \sigma(x)\, dW_t. \tag{2.21}$$

This classical result follows from a second-order Taylor expansion of B together with the Itô multiplication rules; see [131] for a complete proof.

2.4.3 Infinitesimal Generator

The deterministic (drift) part of the Itô expansion defines the *infinitesimal generator* of the diffusion process.

Definition 2.4.2. The infinitesimal generator of the SDE (2.20) applied to a twice continuously differentiable function B is

$$\mathcal{L}B(x,u) = L_f B(x) + L_g B(x) u + \frac{1}{2}\operatorname{tr}\big(\sigma(x)^\top \nabla^2 B(x)\, \sigma(x)\big)\,. \tag{2.22}$$

The generator captures the expected instantaneous rate of change of B along trajectories of the stochastic system. From Itô's lemma (Theorem 2.4.1),

$$\mathbb{E}[dB(x)] = \mathcal{L}B(x,u)\, dt, \tag{2.23}$$

since $\mathbb{E}[\nabla B^\top \sigma\, dW_t] = 0$ by the martingale property of Itô integrals.

Compared to the deterministic Lie derivative $L_f B + L_g B\, u$, the generator includes the additional second-order term $\frac{1}{2}\operatorname{tr}(\sigma^\top \nabla^2 B\, \sigma)$, which accounts for the curvature effects of the stochastic noise.

2.4.4 Stochastic Control Barrier Functions

Definition 2.4.3. A twice continuously differentiable function $B : \mathbb{R}^n \to \mathbb{R}$ is a *Stochastic Control Barrier Function* (SCBF) for the system (2.20) if there exists an extended class-$\mathcal{K}$ function α such that

$$\sup_{u \in \mathbb{R}^m} [\mathcal{L}B(x,u) + \alpha(B(x))] \geq 0 \tag{2.24}$$

for all x in the domain $\mathcal{X}$.

Theorem 2.4.4 (Stochastic Safety [37])**.** *Let B be a SCBF for the system (2.20), and let $u = \mu(x)$ be a Lipschitz continuous controller satisfying $\mathcal{L}B(x,\mu(x)) + \alpha(B(x)) \geq 0$ for all x. If $x(0) \in \mathcal{C} = \{x : B(x) \geq 0\}$, then*

$$\mathbb{P}(B(x(t)) \geq 0 \text{ for all } t \in [0,T]) = 1 \tag{2.25}$$

for any finite horizon $T > 0$, provided the diffusion does not degenerate (i.e., the stochastic integral $\int_0^t \nabla B^\top \sigma \, dW_s$ is a martingale).

Proof. Applying Itô's lemma to $B(x(t))$ yields

$$B(x(t)) = B(x(0)) + \int_0^t \mathcal{L}B(x(s),\mu(x(s)))\, ds + \int_0^t \nabla B(x(s))^\top \sigma(x(s))\, dW_s. \tag{2.26}$$

Using the barrier condition $\mathcal{L}B + \alpha(B) \geq 0$, we obtain

$$\mathcal{L}B(x(s),\mu(x(s))) \geq -\alpha(B(x(s))), \tag{2.27}$$

so

$$B(x(t)) \geq B(x(0)) - \int_0^t \alpha(B(x(s)))\, ds + M(t), \tag{2.28}$$

where $M(t) = \int_0^t \nabla B(x(s))^\top \sigma(x(s))\, dW_s$ is a continuous local martingale with $M(0) = 0$.
Define $y(t) = B(x(t)) - M(t)$, which satisfies $\dot{y}(t) \geq -\alpha(y(t) + M(t))$ in the sense of distributions. By a stochastic comparison argument (extending the deterministic Comparison Lemma to the semimartingale setting [37]), the process $B(x(t))$ is bounded below by the solution of the stochastic differential inequality

$$dZ(t) = -\alpha(Z(t))\, dt + \nabla B(x(t))^\top \sigma(x(t))\, dW_t, \quad Z(0) = B(x(0)) \geq 0. \tag{2.29}$$

Since $\alpha(0) = 0$, the origin $Z = 0$ is an absorbing barrier from above in expectation:

$\mathbb{E}[dZ \mid Z = 0] = 0$ and Z cannot drift below zero faster than the noise pushes it. Under standard regularity conditions (e.g., $\nabla B^\top \sigma$ is bounded, ensuring $M(t)$ is a true martingale), applying the optional stopping theorem shows that $B(x(t)) \geq 0$ almost surely for all $t \in [0, T]$. □

Remark 2.4.5. In practice, the noise may cause rare excursions below the boundary. When the diffusion coefficient σ is state-dependent and potentially large near the boundary, the almost-sure guarantee weakens to a high-probability guarantee. Chapter 3 addresses this by incorporating Lipschitz bounds on the network to derive explicit probability bounds for safety.

2.5 Robust Control Barrier Functions

The preceding section addressed stochastic disturbances modelled as Brownian-motion noise, which enters the dynamics through a diffusion term and admits probabilistic safety guarantees via Itô calculus. In many robotic applications, however, the uncertainty is better described as a *bounded but otherwise unknown deterministic disturbance*—for example, unmodelled aerodynamic drag, friction, payload variation, or an adversarial perturbation. Robust Control Barrier Functions provide *worst-case* safety guarantees for such settings [191, 75].

2.5.1 Systems with Bounded Disturbances

Consider the perturbed control-affine system

$$\dot{x} = f(x) + g(x)u + d(t), \tag{2.30}$$

where $d : [0, \infty) \to \mathcal{D}$ is a measurable disturbance signal taking values in the compact set

$$\mathcal{D} = \{d \in \mathbb{R}^n : \|d\| \leq \bar{d}\} \tag{2.31}$$

for a known bound $\bar{d} > 0$. The drift and input fields f, g and the safe set $\mathcal{C} = \{x : B(x) \geq 0\}$ are defined as in (2.1) and (2.8), respectively.

2.5.2 Definition and Safety Guarantee

A robust CBF must ensure that a safe control exists for *every* admissible disturbance realisation.

Definition 2.5.1 (Robust Control Barrier Function [191, 75])**.** A continuously differentiable function $B : \mathbb{R}^n \to \mathbb{R}$ is a *Robust Control Barrier Function* (RCBF) for (2.30) if there exists an extended class-$\mathcal{K}$ function α such that, for all $x \in \mathbb{R}^n$,

$$\sup_{u \in \mathbb{R}^m} \inf_{d \in \mathcal{D}} \left[L_f B(x) + L_g B(x) u + \nabla B(x)^\top d + \alpha(B(x)) \right] \geq 0. \tag{2.32}$$

The sup–inf order reflects the worst-case perspective: the controller selects u before (or simultaneously with) nature selecting d, and safety must hold regardless of the disturbance. Evaluating the inner infimum in closed form is straightforward. Since $\mathcal{D}$ is a norm ball, the Cauchy–Schwarz inequality gives

$$\inf_{d \in \mathcal{D}} \nabla B(x)^\top d = -\|\nabla B(x)\| \, \bar{d}, \tag{2.33}$$

with the infimum attained at $d^\star = -\bar{d} \nabla B(x) / \|\nabla B(x)\|$ whenever $\nabla B(x) \neq 0$. Substituting (2.33) into (2.32) yields the equivalent scalar condition

$$\sup_{u \in \mathbb{R}^m} \left[L_f B(x) + L_g B(x) u + \alpha(B(x)) - \|\nabla B(x)\| \, \bar{d} \right] \geq 0. \tag{2.34}$$

Theorem 2.5.2 (Robust Safety [191])**.** *If B is a Robust CBF for (2.30), then there exists a Lipschitz continuous controller $u = \mu(x)$ such that the safe set $\mathcal{C}$ is forward invariant for* every *measurable disturbance $d(\cdot)$ with $d(t) \in \mathcal{D}$.*

The proof mirrors the nominal CBF case: the RCBF condition tightens the barrier inequality by the worst-case disturbance margin $\|\nabla B(x)\| \bar{d}$, after which the comparison argument of Lemma 2.1 applies verbatim; see [191].

2.5.3 Robust CBF Quadratic Program

The robust safety-critical controller is obtained by solving

$$u^\star(x) = \arg\min_{u \in \mathbb{R}^m} \|u - u_{\text{ref}}(x)\|^2 \quad \text{subject to} \quad L_f B(x) + L_g B(x) u + \alpha(B(x)) \geq \|\nabla B(x)\| \, \bar{d}, \tag{2.35}$$

which tightens the nominal CBF–QP (2.11) by the margin $\|\nabla B(x)\| \bar{d}$. Intuitively, the controller must maintain a safety margin proportional to both the barrier gradient magnitude and the disturbance bound, ensuring that even the worst-case perturbation cannot push the state out of $\mathcal{C}$.

Remark 2.5.3 (Input-to-State Safety). When the disturbance bound $\bar{d}$ is too large for condition (2.34) to hold—that is, the system cannot fully reject the disturbance—exact forward invariance of $\mathcal{C}$ can no longer be guaranteed. In this regime, Xu et al. [191] introduce the notion of *Input-to-State Safety (ISSf)*: the system remains within an inflated safe set $\mathcal{C}_\gamma = \{x : B(x) \geq -\gamma(\bar{d})\}$ for a class-$\mathcal{K}$ function γ that quantifies the safety degradation as a function of the disturbance magnitude. ISSf is the safety analogue of Input-to-State Stability (ISS) and provides a principled framework for graceful degradation of safety margins.

Remark 2.5.4 (Stochastic vs. Robust Perspectives). Stochastic CBFs and robust CBFs address complementary forms of uncertainty. Stochastic CBFs model zero-mean process noise via Itô calculus and yield probabilistic safety guarantees, whereas robust CBFs handle adversarial, worst-case disturbances and provide deterministic guarantees at the cost of conservatism. Several chapters in this thesis encounter both perspectives: Chapter 5 explicitly models stochastic disturbances, Chapter 4 uses conformal prediction to construct probabilistic safety margins that bridge the two paradigms, and Chapter 7 evaluates controller resilience under external perturbations.

2.6 Dynamic Programing and HJ Formulations

Optimal control provides a complementary viewpoint to barrier functions. Whereas barriers enforce safety by constraining trajectories, value functions quantify the best achievable performance under a cost.

Given dynamics $\dot{x} = f(x) + g(x)u$ and cost functional

$$J(u(\cdot)) = \int_0^T \ell(x(t), u(t))\, dt + \phi(x(T)), \tag{2.36}$$

the value function is defined as

$$V(x,t) = \inf_{u(\cdot)} J(u(\cdot)). \tag{2.37}$$

Theorem 2.6.1 (Hamilton–Jacobi–Bellman)**.** *If the value function V defined by (2.37) is continuously differentiable, then it satisfies*

$$-\frac{\partial V}{\partial t} = \inf_u \left\{ \ell(x,u) + \nabla V(x,t)^\top \big(f(x) + g(x)u\big) \right\}, \tag{2.38}$$

with terminal condition $V(x, T) = \phi(x)$.

This classical result follows from Bellman's principle of optimality via a Taylor expansion of the value function in small time δ; see [52] for a complete derivation.

Remark 2.6.2. When the value function is not differentiable (e.g., due to discontinuities in the optimal control), the HJB equation must be interpreted in the viscosity sense [40]. In this framework, classical derivatives are replaced by test functions, and the PDE is satisfied in a generalized sense. The physics-informed neural network approximation in Chapter 8 sidesteps this issue by enforcing the PDE residual at collocation points, which naturally accommodates non-smooth solutions.

Equation (2.38) forms the basis for the physics-informed approximation schemes developed in Chapter 8.

2.6.1 Hamilton–Jacobi Reachability

Safety verification often requires reasoning about worst-case disturbances. Hamilton–Jacobi (HJ) reachability analysis provides a principled framework for computing the *maximal* set of states from which safety can be guaranteed, even in the presence of bounded adversarial disturbances. Unlike CBFs, which verify a *given* barrier function, HJ reachability *constructs* the value function (and hence the safe set) from scratch via dynamic programming. The price is computational: the method scales exponentially with state dimension [12, 107].

Differential Games and Nonanticipative Strategies

Consider a system with both control and disturbance inputs:

$$\dot{x} = f(x) + g_1(x)u + g_2(x)w, \tag{2.39}$$

where $u \in \mathcal{U}$ is the control, $w \in \mathcal{W}$ is a bounded disturbance, and $\mathcal{U}, \mathcal{W}$ are compact sets. The interaction between the controller and the disturbance is formalised as a two-player zero-sum differential game. The disturbance selects a *nonanticipative strategy* $\xi_w : \mathbb{U} \to \mathbb{W}$—a mapping from control signals to disturbance signals that may depend on the past and present control but not the future [52]. Nonanticipative strategies model a worst-case adversary that reacts instantaneously to the controller's actions.

Backward Reachable Tube and Maximal Safe Set

Fix a terminal time $T > 0$ and an initial time t with $t \leq T$. Let $\mathcal{F} \subset \mathbb{R}^n$ be a closed *failure (unsafe) set*, represented as the sub-zero level set of a Lipschitz function ϕ:

$$\mathcal{F} = \{x \in \mathbb{R}^n : \phi(x) \leq 0\}. \tag{2.40}$$

The function ϕ encodes the safety margin: $\phi(x) > 0$ means the state x is safe, and $\phi(x) \leq 0$ means it violates the safety constraint. The question HJ reachability answers is: *starting from a state x at time t, can the disturbance w force the trajectory into $\mathcal{F}$ at some future time before T, despite the best control effort?*

Definition 2.6.3 (Backward Reachable Tube)**.** Given the failure set $\mathcal{F}$, the terminal time T, and the dynamics (2.39), the *backward reachable tube* at time $t \leq T$ is

$$\mathcal{R}(t) = \{x \in \mathbb{R}^n : \forall\, u(\cdot) \in \mathbb{U},\ \exists\, w(\cdot) \in \mathbb{W},\ \exists\, \tau \in [t, T] \text{ s.t. } x(\tau) \in \mathcal{F},\ x(t) = x\}\,, \tag{2.41}$$

where $\mathbb{U}$ and $\mathbb{W}$ denote the sets of measurable control and disturbance signals on the interval $[t, T]$, respectively.

Three variables parameterise this set; their roles are:

- t is the **initial (query) time**: we ask which states are dangerous if the system starts at time t.
- T is the **terminal time**: the end of the planning horizon. Because $t \leq T$, the interval $[t, T]$ is the future window over which safety must be maintained.
- $\tau \in [t, T]$ is the **violation time**: the (existentially quantified) moment at which the trajectory enters the failure set $\mathcal{F}$.

In words, a state x belongs to $\mathcal{R}(t)$ if, starting from x at time t, *for every* control signal $u(\cdot)$ there *exists* a disturbance signal $w(\cdot)$ and a time $\tau \in [t, T]$ such that the resulting trajectory reaches $\mathcal{F}$ at time τ. The set $\mathcal{R}(t)$ depends on t because a longer remaining horizon (smaller t for fixed T) gives the adversary more time to steer the system into $\mathcal{F}$; consequently, $\mathcal{R}(t)$ is non-decreasing as t decreases:

$$t_1 \leq t_2 \leq T \quad \Longrightarrow \quad \mathcal{R}(t_2) \subseteq \mathcal{R}(t_1). \tag{2.42}$$

At the terminal time itself, $\mathcal{R}(T) = \mathcal{F}$, since the only "future" is the current instant.

Definition 2.6.4 (Maximal Safe Set)**.** The maximal controlled-invariant safe set at time t is the complement of the backward reachable tube:

$$\mathcal{S}(t) = \mathbb{R}^n \setminus \mathcal{R}(t). \tag{2.43}$$

For any initial state $x \in \mathcal{S}(t)$, there exists a control strategy that keeps the trajectory outside $\mathcal{F}$ for the entire interval $[t, T]$, regardless of the disturbance.

The backward reachable tube can be characterized through a value function that satisfies a Hamilton–Jacobi variational inequality (HJ-VI).

Value Function and the HJ-VI

Definition 2.6.5 (Reachability Value Function)**.** Define the value function

$$V(x,t) = \sup_{u(\cdot)\in\mathbb{U}} \inf_{\xi_w\in\Xi} \min_{\tau\in[t,T]} \phi(x(\tau)), \tag{2.44}$$

where $x(\cdot)$ satisfies (2.39) with $x(t) = x$, and Ξ denotes the set of nonanticipative disturbance strategies.

The value function captures the outcome of a differential game between the controller (maximizer, trying to keep $\phi > 0$) and the disturbance (minimizer, trying to drive $\phi \leq 0$). A state x is safe at time t if and only if $V(x,t) > 0$.

Theorem 2.6.6 (HJ Variational Inequality [108, 55])**.** *The value function V defined by (2.44) is the unique Lipschitz continuous* viscosity solution *of the HJ variational inequality*

$$\min\left\{V(x,t) - \phi(x),\ -\frac{\partial V}{\partial t} - H(x, \nabla_x V)\right\} = 0, \tag{2.45}$$

with terminal condition $V(x,T) = \phi(x)$, where the Hamiltonian is

$$H(x,p) = \max_{u\in\mathcal{U}} \min_{w\in\mathcal{W}} \left\{p^\top \big(f(x) + g_1(x)u + g_2(x)w\big)\right\}. \tag{2.46}$$

The derivation follows from the dynamic programming principle applied to the min-max differential game: an obstacle condition $V \leq \phi$ captures the constraint, the HJ equation governs the interior, and their minimum yields the variational inequality; see [108, 55].

Remark 2.6.7 (Viscosity Solutions). The HJ-VI (2.45) generally admits only Lipschitz continuous solutions, not classical (C^1) ones, because the optimal control may switch discontinuously. The notion of *viscosity solution* [40] provides the correct weak solution concept: V satisfies (2.45) if and only if it is simultaneously a viscosity subsolution (tested with smooth functions touching from above) and a viscosity supersolution (touching from below). Under Lipschitz continuity of the dynamics and the target function ϕ, the viscosity solution exists, is unique, and is itself Lipschitz continuous [52]. This regularity is sufficient for defining the gradient $\nabla_x V$ almost everywhere (by Rademacher's theorem), which in turn allows the safe controller to be synthesised at almost every state.

Remark 2.6.8 (Safe Controller Synthesis). The backward reachable tube is recovered as $\mathcal{R}(t) = \{x : V(x,t) \leq 0\}$, and the safe set is $\mathcal{S}(t) = \{x : V(x,t) > 0\}$. The optimal safe controller at each state is obtained from the Hamiltonian:

$$u^*(x) = \arg\max_{u \in \mathcal{U}} \min_{w \in \mathcal{W}} \nabla_x V(x,t)^\top (f(x) + g_1(x)u + g_2(x)w). \tag{2.47}$$

In the interior of the safe set ($V(x,t) > 0$), any control is admissible; the optimal control is needed only near the boundary. A common engineering practice is to apply a least-restrictive controller that switches to u^* only when $V(x,t)$ is close to zero. However, this switching approach can produce undesirable chattering behaviour, motivating the smoother CBVF-based formulation described in Section 2.6.2.

Remark 2.6.9 (Infinite-Horizon Viability). Taking the time horizon to infinity, $V^\infty(x) = \lim_{t \to -\infty} V(x,t)$, yields the infinite-horizon value function whose zero-superlevel set $\mathcal{S}^\infty = \{x : V^\infty(x) \geq 0\}$ is the maximal controlled-invariant set contained in $\{x : \phi(x) \geq 0\}$. In the absence of disturbance, V^∞ satisfies the CBF condition for *any* extended class-$\mathcal{K}_\infty$ function α, since $\dot{V}^\infty \geq 0$ under the optimal control. Hence, setting $B = V^\infty$ yields a valid CBF whose safe set is the largest possible [33]. However, V^∞ is typically not smooth, making direct use as a CBF problematic.

Remark 2.6.10 (Computational Challenge). Solving the HJ-VI on a grid requires $O(N^n)$ computation, where N is the number of grid points per dimension and n is the state dimension. This "curse of dimensionality" limits classical grid-based methods (e.g., Level Set Toolbox [107]) to systems with $n \leq 5$. This computational barrier is the primary motivation for the physics-informed neural network surrogates developed in Chapters 8 and 9, which parameterize V as a neural network and enforce the HJ-VI via collocation losses, scaling to systems with $n \geq 10$. DeepReach [13] demonstrated that such neural approximations can accurately

capture the zero level set even for systems with $n \geq 9$, opening the door to high-dimensional safety verification.

2.6.2 Control Barrier-Value Functions: Unifying CBFs and HJ Reachability

The HJ reachability value function V and Control Barrier Functions B attack the same safety problem from complementary angles. HJ reachability is *constructive*—it produces the maximal safe set—but its optimal controller is overly conservative (it never allows the value to decrease, cf. Remark above). CBFs produce smooth safety-filtering controllers via the QP (2.11), but finding a valid CBF is itself a hard problem. Choi et al. [33] unify the two by introducing the *Control Barrier-Value Function* (CBVF).

Definition 2.6.11 (Robust Control Barrier-Value Function [33])**.** For a discount rate $\gamma \geq 0$, the Robust CBVF is defined as

$$B_\gamma(x,t) = \inf_{\xi_w \in \Xi} \sup_{u(\cdot)\in\mathbb{U}} \min_{\tau\in[t,0]} e^{\gamma(\tau-t)}\, \phi(x(\tau)), \tag{2.48}$$

where $x(\cdot)$ solves (2.39) with $x(t) = x$, and ϕ is the safety target function as before.

The exponential weighting $e^{\gamma(\tau-t)}$ relaxes the reachability cost by allowing the safety margin to decay at rate γ backward in time. When $\gamma = 0$, the CBVF reduces to the standard reachability value function V in (2.44).

Theorem 2.6.12 (CBVF Variational Inequality [33])**.** *The CBVF B_γ is the unique Lipschitz continuous viscosity solution of*

$$\min\left\{\phi(x) - B_\gamma(x,t),\ \ \frac{\partial B_\gamma}{\partial t} + \max_{u\in\mathcal{U}} \min_{w\in\mathcal{W}} \nabla_x B_\gamma^\top (f + g_1 u + g_2 w) + \gamma B_\gamma(x,t)\right\} = 0, \tag{2.49}$$

with terminal condition $B_\gamma(x,0) = \phi(x)$.

The key observation is that the second branch of (2.49) is precisely the robust CBF condition: if the state is in the interior of the safe set ($B_\gamma < \phi$), then the CBVF must satisfy

$$\dot{B}_\gamma + \gamma B_\gamma \geq 0 \qquad \Longleftrightarrow \qquad \dot{B}_\gamma \geq -\gamma B_\gamma, \tag{2.50}$$

which is the CBF decay condition with a linear class-$\mathcal{K}_\infty$ function $\alpha(z) = \gamma z$.

Proposition 2.6.13 (Safety Equivalence [33]). *The zero-superlevel set of B_γ coincides with the viability kernel: $\{x : B_\gamma(x,t) \geq 0\} = \mathcal{S}(t)$ for all $\gamma \geq 0$ and all $t \leq 0$.*

Thus, B_γ recovers the maximal safe set (like the standard reachability value function) while also satisfying a CBF-like decay condition (unlike the standard value function, which enforces $\dot{V} \geq 0$). This enables a *Robust CBVF-QP* for control-affine systems:

$$u^\star(x,t) = \arg\min_{u \in \mathcal{U}} \|u - u_{\text{ref}}\|^2 \quad \text{s.t.} \quad \frac{\partial B_\gamma}{\partial t} + \min_{w \in \mathcal{W}} \nabla_x B_\gamma^\top (f + g_1 u + g_2 w) + \gamma B_\gamma \geq 0, \tag{2.51}$$

which is always feasible inside $\mathcal{S}(t)$ and strictly less conservative than the standard reachability controller for $\gamma > 0$: it permits the value function to decrease (at rate γ) when the state is safely in the interior, granting more control authority for task execution.

Remark 2.6.14. The CBVF framework provides the theoretical foundation for Chapter 7, where a Neural CBVF is learned by combining offline value function estimation with barrier function training. The CBVF-QP serves as the deployment-time safety filter.

2.7 Physics-Informed Neural Networks

The curse of dimensionality renders grid-based PDE solvers impractical for systems with more than approximately five state dimensions (Remark 2.6.7). Physics-Informed Neural Networks (PINNs) [127] offer a mesh-free alternative by parameterising the PDE solution as a neural network and training it to satisfy the governing equation at randomly sampled *collocation points*. This section introduces the general PINN framework, the role of automatic differentiation in computing exact derivatives, and the sinusoidal (SIREN) activation functions that are particularly well-suited for representing solutions of PDEs arising in reachability and optimal control.

2.7.1 General Framework

Consider a PDE of the form

$$\mathcal{N}[u](t,x) = 0, \quad (t,x) \in \Omega, \tag{2.52}$$

with boundary/initial conditions $\mathcal{B}[u](t,x) = 0$ on $\partial\Omega$, where $\mathcal{N}$ is a (possibly nonlinear) differential operator and u is the unknown solution. PINNs approximate u by a neural network $u_\theta(t,x)$ with parameters θ and define the training loss as a weighted sum of the PDE residual and boundary condition errors:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_r}\sum_{i=1}^{N_r} \|\mathcal{N}[u_\theta](t_i,x_i)\|^2}_{\text{PDE residual loss}} + \lambda \underbrace{\frac{1}{N_b}\sum_{j=1}^{N_b} \|\mathcal{B}[u_\theta](t_j,x_j)\|^2}_{\text{boundary condition loss}}, \tag{2.53}$$

where $\{(t_i,x_i)\}_{i=1}^{N_r}$ are collocation points sampled from the interior of Ω, $\{(t_j,x_j)\}_{j=1}^{N_b}$ are points on $\partial\Omega$, and $\lambda > 0$ controls the relative importance of the two terms.

Remark 2.7.1. The PINN loss (2.53) provides a *self-supervised* learning signal: no ground-truth solution is needed for training. The physics of the problem is encoded entirely through the differential operator $\mathcal{N}$ and the boundary operator $\mathcal{B}$. This is in contrast to standard supervised regression, where input–output pairs from a known solution are required.

A key advantage of PINNs over classical numerical methods is their *mesh-free* nature: collocation points can be sampled randomly from the domain without constructing a grid, and the computational cost scales with the number of network parameters rather than exponentially with the state dimension. This property makes PINNs particularly attractive for the high-dimensional HJB PDEs arising in optimal control and reachability (Chapters 8 and 9).

2.7.2 Automatic Differentiation for PDE Derivatives

Evaluating the PDE residual $\mathcal{N}[u_\theta]$ requires computing partial derivatives of the network output with respect to its inputs (e.g., $\partial_t u_\theta$, $\nabla_x u_\theta$). PINNs compute these derivatives *exactly* using automatic differentiation (AD) [15], which propagates derivative information through the computational graph of the neural network via the chain rule.

Definition 2.7.2 (Automatic Differentiation)**.** Let $u_\theta : \mathbb{R}^{n+1} \to \mathbb{R}$ be a neural network composed of differentiable layers $u_\theta = f_L \circ f_{L-1} \circ \cdots \circ f_1$. *Reverse-mode automatic differentiation* computes the gradient $\nabla_{(t,x)} u_\theta$ by recursively applying the chain rule from the output layer backward:

$$\frac{\partial u_\theta}{\partial x_k} = \frac{\partial f_L}{\partial f_{L-1}} \cdot \frac{\partial f_{L-1}}{\partial f_{L-2}} \cdots \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial x_k}, \tag{2.54}$$

at a computational cost proportional to a single forward pass, regardless of the input dimension.

Remark 2.7.3. Automatic differentiation is distinct from both *numerical differentiation* (finite differences) and *symbolic differentiation*. Finite differences introduce discretisation error proportional to the step size h and are subject to catastrophic cancellation for small h. Symbolic differentiation produces exact expressions but suffers from expression swell for deep networks. AD computes derivatives to machine precision without expression growth, making it ideally suited for enforcing PDE constraints in PINNs.

In the context of HJB PDEs, automatic differentiation is used to compute the temporal derivative $\partial_t u_\theta$ and the spatial gradient $\nabla_x u_\theta$ that appear in the Hamiltonian. Since these derivatives are computed exactly through the network's computational graph, the PDE residual loss involves no discretisation error—a significant advantage over finite-difference schemes that require careful grid design to maintain accuracy.

2.7.3 Sinusoidal Representation Networks (SIREN)

The choice of activation function significantly impacts a PINN's ability to represent PDE solutions and produce accurate derivatives. Standard activations such as ReLU ($\sigma(x) = \max(0, x)$) are piecewise linear: their second derivative is zero almost everywhere, making them poorly suited for PDEs involving higher-order derivatives. Smooth activations such as tanh or sigmoid avoid this issue but can suffer from vanishing gradients in deep networks.

Sinusoidal Representation Networks (SIREN) [140] address both limitations by using periodic activation functions of the form

$$\sigma(x) = \sin(\omega_0\, x), \tag{2.55}$$

where $\omega_0 > 0$ is a frequency hyperparameter (typically $\omega_0 = 30$). A SIREN layer maps an input vector $\mathbf{z}^{(l-1)} \in \mathbb{R}^{d_{l-1}}$ to an output $\mathbf{z}^{(l)} \in \mathbb{R}^{d_l}$ via

$$\mathbf{z}^{(l)} = \sin\Big(\omega_0 \left(W^{(l)}\mathbf{z}^{(l-1)} + b^{(l)}\right)\Big), \tag{2.56}$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{d_l}$ are the weight matrix and bias vector, respectively.

Desirable Properties for PDE Solving. SIREN activations possess three properties that make them particularly effective for physics-informed learning:

1. **Infinite differentiability.** Since $\sin(\cdot)$ is C^∞, all derivatives of the network output with respect to its inputs exist and are themselves sinusoidal functions. This ensures that the PDE residual and its gradients are well-defined everywhere, enabling stable training with higher-order PDE constraints.

2. **Rich frequency content.** The periodic nature of the activation allows the network to represent high-frequency spatial and temporal variations in the PDE solution without requiring an excessively large number of hidden units.

3. **Derivative stability.** Unlike tanh, whose derivatives saturate for large inputs, the derivatives of $\sin(\omega_0 x)$ remain bounded ($|\cos(\omega_0 x)| \leq 1$), avoiding vanishing gradient problems even in deep architectures.

Weight Initialization. A critical component of SIREN is a specialised weight initialisation scheme that ensures the distribution of activations is preserved across layers. For hidden layers, the weights are initialised uniformly as

$$W_{ij}^{(l)} \sim \mathcal{U}\left[-\frac{\sqrt{6/d_{l-1}}}{\omega_0},\ \frac{\sqrt{6/d_{l-1}}}{\omega_0}\right], \tag{2.57}$$

where d_{l-1} is the fan-in of the layer. This ensures that the pre-activation values $\omega_0(W^{(l)}\mathbf{z}^{(l-1)} + b^{(l)})$ have a distribution that spans $[-\pi, \pi]$ uniformly, keeping the $\sin(\cdot)$ activations in their full dynamic range. The first layer uses a wider initialisation $W_{ij}^{(1)} \sim \mathcal{U}[-1/d_0,\ 1/d_0]$ to accommodate the scale of the raw input coordinates.

> *Remark* 2.7.4. DeepReach [13] employs SIREN networks to approximate the value function of HJ variational inequalities, achieving accurate zero-level-set reconstruction for systems up to 9 dimensions. The combination of sinusoidal activations and automatic differentiation enables exact computation of the Hamiltonian $H(x, \nabla_x V)$ in the PDE residual loss, without any finite-difference approximation. This architecture choice is adopted in Chapters 8 and 9 for learning value functions of HJB PDEs in systems up to 20 dimensions.

2.7.4 Training Challenges and Mitigation Strategies

Despite their flexibility, PINNs face several training challenges that must be addressed for reliable convergence:

Loss Balancing. The PDE residual loss and boundary condition loss in (2.53) can have vastly different magnitudes, causing gradient dominance by one term and

under-fitting of the other. Adaptive loss re-balancing schemes, such as learning-rate annealing based on the gradient statistics of each loss term [181], dynamically adjust the weight λ during training to equilibrate the gradient contributions of different loss components.

Causal Training and Curriculum Learning. For time-dependent PDEs that propagate information in one direction (e.g., backward in time for HJB equations), training the network over the entire spatiotemporal domain simultaneously can lead to poor convergence. The PDE residual at time t depends on the accuracy of the solution at later times, which the network has not yet learned. *Curriculum learning* addresses this by progressively expanding the temporal training window, starting from the terminal condition and gradually including earlier times. This strategy, introduced in the DeepReach framework [13], respects the causal structure of the PDE and significantly improves convergence in practice.

Spectral Bias. Standard neural networks tend to learn low-frequency components of the target function first, a phenomenon known as *spectral bias* or the *frequency principle*. This can delay or prevent convergence to PDE solutions with sharp gradients or discontinuities. SIREN activations mitigate this bias by providing the network with inherent high-frequency representational capacity through the periodic activation function.

> *Remark* 2.7.5. The training strategies discussed above—adaptive loss balancing, curriculum learning, and SIREN activations—are all employed in the physics-informed framework developed in Chapter 8. Together, they enable reliable training of neural PDE solvers for high-dimensional Hamilton–Jacobi equations arising in safety-critical optimal control.

2.8 Reinforcement Learning and Markov Decision Processes

Reinforcement learning (RL) provides a data-driven framework for sequential decision making in uncertain environments. An RL problem is typically formulated as a Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{X}, \mathcal{U}, P, r, \gamma), \tag{2.58}$$

where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, P is the transition kernel, r is the reward function, and $\gamma \in (0,1)$ is the discount factor.

A policy $\pi : \mathcal{X} \to \mathcal{U}$ induces a value function defined by

$$V^{\pi}(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \,\middle|\, x_0 = x,\ u_t = \pi(x_t)\right]. \tag{2.59}$$

Optimality is defined via $V^{\star}(x) = \sup_{\pi} V^{\pi}(x)$.

Definition 2.8.1. The Bellman operator associated with a policy π is

$$(\mathcal{T}^{\pi} V)(x) = r(x, \pi(x)) + \gamma \mathbb{E}[V(x') \mid x, \pi(x)]\,. \tag{2.60}$$

Theorem 2.8.2 (Contraction of Bellman Operators [122])**.** *The operator $\mathcal{T}^{\pi}$ is a γ-contraction in the supremum norm:*

$$\|\mathcal{T}^{\pi} V_1 - \mathcal{T}^{\pi} V_2\|_{\infty} \leq \gamma \|V_1 - V_2\|_{\infty}. \tag{2.61}$$

Consequently, $\mathcal{T}^{\pi}$ has a unique fixed point V^{π}.

This is the classical contraction-mapping argument: the reward terms cancel, Jensen's inequality and the supremum norm yield the γ-factor, and Banach's fixed-point theorem gives uniqueness of V^{π} with geometric convergence of value iteration; see [122].

This contraction guarantees convergence of value iteration and plays a critical role in the offline reinforcement learning methods used to shape barrier functions in Chapter 7.

2.8.1 Action-Value Functions and Q-Learning

In many practical settings, the *action-value* (Q) function is more useful than the state-value function because it allows policy improvement without a model of the dynamics.

Definition 2.8.3 (Action-Value Function)**.** The action-value function under policy π is

$$Q^{\pi}(x, u) = r(x, u) + \gamma\, \mathbb{E}[V^{\pi}(x') \mid x, u]\,. \tag{2.62}$$

The optimal action-value function is $Q^{*}(x, u) = \sup_{\pi} Q^{\pi}(x, u)$, and the optimal

policy satisfies $\pi^*(x) = \arg\max_u Q^*(x,u)$.

The optimal Bellman equation for Q^* is

$$Q^*(x,u) = r(x,u) + \gamma\, \mathbb{E}\Big[\max_{u'} Q^*(x',u') \;\Big|\; x,u\Big]. \tag{2.63}$$

Q-learning [182] approximates Q^* by iteratively applying stochastic samples of the optimal Bellman operator, converging under standard stochastic approximation conditions. Deep Q-Networks (DQN) [109] parameterise Q as a neural network and stabilise training through experience replay and target networks, but are limited to discrete action spaces.

2.8.2 Policy Gradient Methods

For continuous action spaces, policy gradient methods parameterise the policy directly as π_θ and optimise the expected return $J(\theta) = \mathbb{E}_{x_0}[V^{\pi_\theta}(x_0)]$ by gradient ascent.

Theorem 2.8.4 (Policy Gradient Theorem [150]). *Let π_θ be a differentiable parameterised policy with parameters θ. Then*

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim d^{\pi_\theta},\, u\sim\pi_\theta}[Q^{\pi_\theta}(x,u)\, \nabla_\theta \log \pi_\theta(u \mid x)]\,, \tag{2.64}$$

where $d^{\pi_\theta}(x) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(x_t = x \mid \pi_\theta)$ is the discounted state distribution.

Remark 2.8.5 (Variance Reduction). In practice, the action-value function Q^{π_θ} in (2.64) is replaced by an *advantage function* $A^{\pi_\theta}(x,u) = Q^{\pi_\theta}(x,u) - V^{\pi_\theta}(x)$, which does not change the expected gradient but substantially reduces variance.

2.8.3 Actor–Critic Architectures

Actor–critic methods combine the policy gradient approach (actor) with a learned value function (critic) for lower-variance gradient estimates:

- **Critic:** A value function $V_\omega(x) \approx V^{\pi_\theta}(x)$ is learned by minimising the temporal-difference (TD) error

$$\delta_t = r_t + \gamma V_\omega(x_{t+1}) - V_\omega(x_t), \tag{2.65}$$

  i.e., ω is updated to reduce $\mathbb{E}[\delta_t^2]$.

- **Actor:** The policy parameters θ are updated using the estimated advantage $\hat{A}_t = \delta_t$ as a substitute for A^{π_θ} in (2.64).

Proximal Policy Optimization (PPO) [137] introduces a clipped surrogate objective that constrains the policy update magnitude, ensuring stable training. Soft Actor–Critic (SAC) [64] adds an entropy regularization term to the objective, encouraging exploration while maintaining near-optimal performance:

$$J_{\mathrm{SAC}}(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \Big(r(x_t, u_t) + \alpha_{\mathrm{ent}}\, \mathcal{H}(\pi_\theta(\cdot \mid x_t)) \Big)\right], \tag{2.66}$$

where $\mathcal{H}$ denotes entropy and $\alpha_{\mathrm{ent}} > 0$ is a temperature parameter. SAC is the base RL algorithm used in Chapter 10 for training visuomotor policies that are then filtered by a safety module.

2.8.4 Safety in Reinforcement Learning

Classical reinforcement learning focuses on maximizing long-term reward but does not, by default, maintain safety during learning or deployment. Safety constraints can be introduced at several levels: through constrained MDP formulations, by incorporating penalties in the reward signal, via model-based safety checks that predict and prevent future violations, or by integrating CBF-based safety filters that override unsafe actions in real time. Most real-world systems, however, require *zero violation* safety, which aligns naturally with control-theoretic certificate approaches. The integration of RL and barrier functions is a central theme of Chapter 7.

2.8.5 Constrained Markov Decision Processes

Constrained MDPs (CMDPs) provide the standard formalization for sequential decision making under safety constraints [5]. A CMDP augments the standard MDP with one or more auxiliary cost functions and constraint thresholds.

Definition 2.8.6. A *Constrained Markov Decision Process* is a tuple

$$\mathcal{M}_c = (\mathcal{X},\, \mathcal{U},\, P,\, r,\, \gamma,\, \{c_i\}_{i=1}^{K},\, \{d_i\}_{i=1}^{K}), \tag{2.67}$$

where $(\mathcal{X}, \mathcal{U}, P, r, \gamma)$ define a standard MDP, each $c_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ is a constraint cost function, and $d_i \geq 0$ is the corresponding cost threshold.

A policy π induces a *constraint return* for each cost:

$$J_{c_i}(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c_i(x_t, u_t) \;\middle|\; u_t = \pi(x_t)\right]. \tag{2.68}$$

The CMDP objective is to find the policy that maximizes expected reward while satisfying every constraint:

$$\pi^* = \arg\max_{\pi} \; V^{\pi}(x_0) \quad \text{subject to} \quad J_{c_i}(\pi) \leq d_i, \quad i = 1, \ldots, K. \tag{2.69}$$

Theorem 2.8.7 (Existence of Optimal CMDP Policies [5])**.** *For a CMDP with a finite or compact action space and K constraint costs, there exists an optimal policy π^* that solves (2.69). Moreover, π^* can be chosen to be a mixture of at most $K+1$ deterministic stationary policies.*

The proof uses the occupation-measure reformulation: the achievable occupation measures form a convex polytope on which the reward and constraint returns are linear, so the CMDP reduces to a linear program whose optimum is attained at a vertex (a mixture of at most $K+1$ deterministic stationary policies, by Carathéodory's theorem); see [5].

Lagrangian Relaxation. In practice, the constrained optimization (2.69) is often solved via Lagrangian relaxation. The Lagrangian is

$$\mathcal{L}(\pi, \lambda) = V^{\pi}(x_0) - \sum_{i=1}^{K} \lambda_i \left(J_{c_i}(\pi) - d_i\right), \tag{2.70}$$

where $\lambda_i \geq 0$ are Lagrange multipliers. Under strong duality (which holds for CMDPs [5]), the CMDP solution is obtained via the saddle-point problem

$$\pi^* = \arg\max_{\pi} \min_{\lambda \geq 0} \; \mathcal{L}(\pi, \lambda). \tag{2.71}$$

Algorithms such as Constrained Policy Optimization (CPO) [3] and PPO-Lagrangian approximate this saddle-point by alternating between policy gradient updates and dual variable updates. However, these methods enforce constraints only *in expectation* and *on average*, which may still permit individual trajectories to violate safety constraints.

Remark 2.8.8. CMDPs treat safety as a *soft* constraint: the expected cumulative cost must remain below a threshold, but instantaneous violations are permitted.

In contrast, Control Barrier Functions (Section 2.3) enforce *hard* pointwise constraints that prevent any trajectory from leaving the safe set. This fundamental difference motivates the integration of CBF-based safety filters with RL policies developed in Chapters 7 and 10: the RL policy optimizes reward subject to the CMDP objective, while the CBF filter provides a hard safety backstop that overrides unsafe actions.

2.8.6 Offline RL and Distribution Shift

Offline RL concerns itself with learning a policy from a fixed dataset

$$\mathcal{D} = \{(x_i, u_i, r_i, x_i')\}_{i=1}^{N} \tag{2.72}$$

collected by an unknown behavior policy. A core challenge is the *distribution shift* between the dataset and the learned policy.

Definition 2.8.9. A distribution shift occurs when the state–action pairs visited by the learned policy lie outside the support of the dataset $\mathcal{D}$.

This phenomenon leads to *extrapolation error*: estimates of value or model dynamics are unreliable outside the data distribution.

Proposition 2.8.10. *If a value function estimator $\widehat{V}$ has bias $\epsilon(x,u)$ at out-of-distribution state–action pairs, i.e., $\widehat{V}(x) = V^{\pi}(x) + \epsilon(x, \pi(x))$, then the greedy policy $\hat{\pi}(x) = \arg\max_u \left[r(x,u) + \gamma \mathbb{E}[\widehat{V}(x') \mid x, u]\right]$ may select actions that are arbitrarily suboptimal or unsafe.*

Proof. Consider the greedy policy improvement step. The greedy policy selects

$$\hat{\pi}(x) = \arg\max_u \left\{ r(x,u) + \gamma \mathbb{E}\left[\widehat{V}(x') \mid x, u\right] \right\}. \tag{2.73}$$

Substituting $\widehat{V}(x') = V^{\pi}(x') + \epsilon(x', \hat{\pi}(x'))$:

$$\hat{\pi}(x) = \arg\max_u \left\{ r(x,u) + \gamma \mathbb{E}\left[V^{\pi}(x') + \epsilon(x', \hat{\pi}(x')) \mid x, u\right] \right\}. \tag{2.74}$$

The bias term ϵ is uncontrolled outside the data distribution. If the estimator $\widehat{V}$ is overoptimistic at some out-of-distribution states (i.e., $\epsilon(x', u') \gg 0$ for some (x', u')), then the greedy operator preferentially selects actions u that steer the system toward those states.
Formally, let (x_0, u_0) be outside $\mathrm{supp}(\mathcal{D})$ with $\epsilon(x_0, u_0) = M$ for some $M > 0$. The

greedy policy error satisfies

$$V^{\hat{\pi}}(x) - V^{\pi}(x) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\epsilon(x_t, \hat{\pi}(x_t))\right] \leq \frac{\|\epsilon\|_\infty}{1-\gamma}. \tag{2.75}$$

Since $\|\epsilon\|_\infty$ is unbounded when the estimator has no supervision outside $\mathcal{D}$, the performance gap $|V^{\hat{\pi}}(x) - V^*(x)|$ can be made arbitrarily large. In the context of safety, actions selected based on overestimated values may lead the system into unsafe regions where no corrective data exists in $\mathcal{D}$, causing safety violations. □

Techniques such as conservative value estimation and uncertainty penalization address this issue. In Chapter 7, offline value functions are used not to generate direct policies but to sculpt optimal control barrier functions, alleviating distributional errors.

2.9 Conformal Prediction for Safety and Verification

Conformal prediction provides a framework for producing valid uncertainty sets with distribution-free guarantees. Let $\widehat{f}$ be a predictor and let (x_i, y_i) denote calibration data.

Definition 2.9.1. The nonconformity score associated with a datapoint is

$$s_i = |y_i - \widehat{f}(x_i)|. \tag{2.76}$$

For a desired confidence level $1-\alpha$, let q_α be the $(1-\alpha)$ empirical quantile of the calibration scores.

Theorem 2.9.2 (Coverage of Conformal Prediction [175, 10])**.** *Assume the calibration data $(x_1, y_1), \ldots, (x_n, y_n)$ and the test point (x_{n+1}, y_{n+1}) are exchangeable. Let $s_i = |y_i - \widehat{f}(x_i)|$ be the nonconformity scores and define q_α as the $\lceil (1-\alpha)(n+1) \rceil / n$ empirical quantile of $\{s_1, \ldots, s_n\}$. Then*

$$\mathbb{P}\Big(y_{n+1} \in \Big[\widehat{f}(x_{n+1}) - q_\alpha,\ \widehat{f}(x_{n+1}) + q_\alpha\Big]\Big) \geq 1 - \alpha. \tag{2.77}$$

The proof relies only on the symmetry implied by exchangeability: the scores $s_1, \ldots, s_{n+1}$ are exchangeable, so the rank of s_{n+1} among them is uniform on $\{1, \ldots, n+1\}$, whence $\mathbb{P}(s_{n+1} \leq q_\alpha) = \lceil (1-\alpha)(n+1) \rceil / (n+1) \geq 1-\alpha$; see [175, 10].

Remark 2.9.3. The coverage guarantee is *marginal*: it holds in expectation over the random draw of (x_{n+1}, y_{n+1}). Conditional coverage — holding for each x_{n+1} separately — requires additional assumptions or adaptive methods. The guarantee is also *distribution-free*: no parametric assumptions on the data-generating process are needed beyond exchangeability. In Chapter 4, this property is leveraged to construct prediction regions for Neural CBF errors that hold with guaranteed coverage.

In later chapters, conformal prediction is used to calibrate both Neural CBFs and neural value functions. This allows constructing safety certificates that hold with finite-sample, distribution-free confidence guarantees.

Remark 2.9.4. Conformal prediction is particularly effective in deterministic settings. Extensions to stochastic systems require delicate handling of temporal dependencies, a topic explored in Chapter 3.

2.10 Latent Representations and Visuomotor Control

Many robotic systems rely on high-dimensional sensory inputs such as images. To integrate these signals into safety-critical control pipelines, one must extract compact latent representations that preserve task-relevant geometry.

Let o_t denote an observation (e.g., an RGB image), and define a latent encoding

$$z_t = \phi(o_t), \tag{2.78}$$

where ϕ is typically a convolutional neural network or a vision transformer.

2.10.1 Latent Dynamics and Control

If the latent representation is sufficiently expressive, one may model latent dynamics as

$$z_{t+1} \approx F_z(z_t, u_t), \tag{2.79}$$

or, in continuous time,

$$\dot{z} = f_z(z) + g_z(z)u. \tag{2.80}$$

Safety constraints are then imposed in latent space via a barrier $B(z)$.

Definition 2.10.1. A latent-space barrier function is a differentiable function $B : \mathbb{R}^d \to \mathbb{R}$ such that the set

$$\mathcal{C}_z = \{z : B(z) \geq 0\} \tag{2.81}$$

approximates the projection of the true safe set.

Remark 2.10.2. Latent safety requires consistency between the geometry of the latent representation and the physical environment. Conformal tools and smoothness regularization help ensure that latent-space certificates correspond to genuine physical safety guarantees.

Visuomotor safety, developed in Chapter 10, builds upon these ideas by combining neural perception with barrier-based safety filters.

2.10.2 World Models

World models learn a *predictive* internal representation of the environment, enabling an agent to simulate future trajectories without interacting with the real system [63]. Formally, a world model consists of three components:

Definition 2.10.3. A *world model* is a tuple (ϕ, F_z, D) where

- $\phi : \mathcal{O} \to \mathbb{R}^d$ is an *encoder* that maps high-dimensional observations o_t to a compact latent state $z_t = \phi(o_t)$,
- $F_z : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ is a *latent dynamics model* predicting $\hat{z}_{t+1} = F_z(z_t, u_t)$,
- $D : \mathbb{R}^d \to \mathcal{O}$ is a *decoder* that reconstructs observations from latent states, $\hat{o}_t = D(z_t)$.

The encoder and dynamics model are typically trained jointly by minimizing a combination of reconstruction loss and latent prediction loss:

$$\mathcal{L}_{\text{wm}} = \underbrace{\sum_t \|o_t - D(\phi(o_t))\|^2}_{\text{reconstruction}} + \beta \underbrace{\sum_t \|\phi(o_{t+1}) - F_z(\phi(o_t), u_t)\|^2}_{\text{latent prediction}}, \tag{2.82}$$

where $\beta > 0$ balances the two objectives. Variational formulations replace the reconstruction term with an evidence lower bound (ELBO) and introduce a KL-divergence regularizer on the latent distribution [65].

Recurrent and Stochastic World Models. To capture partial observability and temporal dependencies, recurrent state-space models (RSSMs) [65] augment the latent state with a deterministic recurrent component h_t and a stochastic component s_t:

$$h_t = \text{GRU}(h_{t-1},\, s_{t-1},\, u_{t-1}), \tag{2.83}$$

$$s_t \sim p(s_t \mid h_t), \tag{2.84}$$

where GRU denotes a Gated Recurrent Unit. The combined state $z_t = (h_t, s_t)$ captures both long-term memory (through h_t) and aleatoric uncertainty (through s_t). The Dreamer family of agents [65, 66] uses this architecture to learn policies entirely from imagined rollouts within the world model, achieving strong performance on visual control benchmarks.

World Models and Safety. World models are particularly relevant to safety-critical visuomotor control for two reasons:

1. **Latent safety certificates.** If the latent dynamics F_z are sufficiently accurate, barrier certificates constructed in the latent space (Section 2.10) transfer to the physical system. The world model provides the latent dynamics needed to evaluate the barrier decrease condition $B(F_z(z_t, u_t)) \leq B(z_t)$.

2. **Safe data augmentation.** Imagined trajectories from the world model can augment the training dataset for the barrier certificate, improving coverage of safety-critical regions that are rarely visited in real data.

> *Remark* 2.10.4. The fidelity of latent safety guarantees depends critically on the accuracy of F_z. Model error in the latent dynamics can cause a latent-space barrier to become invalid in the physical system. Chapter 10 addresses this through the SaLaD framework, which co-trains the latent representation and barrier certificate with a consistency loss that encourages the learned dynamics to preserve safety-relevant structure. Conformal prediction (Section 2.9) provides an additional tool for quantifying model prediction uncertainty with distribution-free guarantees.

2.11 Summary of Background

This chapter has developed the mathematical and statistical foundations necessary for the learning-enabled safety mechanisms explored throughout this dissertation. We began with control-affine dynamical systems, comparison functions, Lie derivatives,

and Lyapunov stability theory, establishing the analytic language used in every subsequent chapter. Control Barrier Functions, including their higher-order, stochastic, and robust variants, were introduced as the primary tool for certifying forward invariance through pointwise derivative conditions and quadratic program-based safety filters.

The optimal control and Hamilton–Jacobi reachability framework provides a complementary, constructive approach: the HJ variational inequality characterises the maximal safe set via viscosity solutions to a min-max differential game. The Control Barrier-Value Function unifies these two perspectives, recovering the maximal safe set while admitting smooth CBF-like controllers via the CBVF-QP. Viability theory ties these concepts together through the notion of controlled invariance.

Physics-informed neural networks provide a scalable, mesh-free approach to solving the high-dimensional PDEs that arise in HJ reachability and optimal control. By embedding the PDE residual into the training loss and computing derivatives exactly through automatic differentiation, PINNs overcome the curse of dimensionality that limits grid-based solvers. Sinusoidal (SIREN) activations further enhance their effectiveness by providing infinite differentiability and rich frequency content.

On the learning side, reinforcement learning (policy gradient methods, actor–critic architectures, and constrained MDPs) provides the data-driven mechanism for acquiring value functions and task structure, while offline RL addresses the distribution-shift challenge that arises when learning from fixed datasets. Conformal prediction supplies distribution-free uncertainty quantification, and latent-space representations with world models extend safety-critical control to high-dimensional sensory inputs.

The synthesis of these theoretical and computational tools forms the backbone of the safe, adaptive, and performance-driven control frameworks developed in the chapters that follow.

With these mathematical and computational foundations in place, we now turn to the central question of this dissertation: how to *learn* safety certificates that come with formal guarantees.

Part I

Neural Control Barrier Functions

This part develops learning-based approaches to synthesize verified Control Barrier Functions. We begin with Lipschitz-based verification for deterministic systems, then introduce distribution-free conformal prediction as a scalable alternative, and finally develop a verification-in-the-loop synthesis framework that extends certificates to stochastic systems and deeper ReLU architectures.

Chapter 3

Lipschitz-Verified Neural Control Barrier Functions

3.1 Related Works

Constructing Control Barrier Functions by hand requires domain expertise and does not scale to complex safety specifications. Neural networks offer a natural solution through their universal approximation capability, but learning a barrier function from data introduces a fundamental challenge: how to *verify* that the learned conditions hold over the *entire* continuous state space, not just at training samples. This chapter addresses this challenge through Lipschitz-based verification for deterministic control-affine systems.

Two settings studied later in this thesis make the need for learned certificates concrete. In the high-dimensional MuJoCo tasks of Chapter 7, the safe set is operationally available to the learner only through labels generated from the simulator's task-specific safety definitions, rather than through a supplied analytical barrier function. In the vision-based setting of Chapter 10, the controller receives images and the certificate is constructed in a learned latent space; without an analytical map from observations to the physical state, an analytical observation-space barrier is not available to design by hand. These cases motivate learning certificates beyond benchmark systems for which physical intuition may suggest an explicit candidate.

The idea of parameterizing CBFs using neural networks, or Neural Control Barrier Functions (NCBFs), was introduced to overcome the difficulty of hand-designing barrier functions for complex systems [44]. Several training procedures have been proposed, including counterexample-guided synthesis [145], SMT-based verification [1],

and differentiable QP layers [124]. However, these methods typically verify the barrier conditions only at finitely many training samples, leaving open the question of whether the conditions hold over the entire continuous state space.

A fundamental challenge with NCBFs is verifying that the learned barrier conditions hold over the *entire* continuous state space, not just at training samples. Approaches based on Satisfiability Modulo Theories (SMT) solvers [1] provide exact verification but scale poorly beyond low-dimensional systems. Lipschitz-based arguments [53, 117] offer an alternative: by bounding the Lipschitz constant of the neural network and its derivatives, one can propagate worst-case errors from finitely many samples to the entire state space. However, constructing the required dense cover still scales poorly with state dimension; the approach is therefore best suited to low- and moderate-dimensional settings, motivating the scalable alternatives developed in later chapters.

Existing NCBF methods learn barrier functions from data but do not provide formal verification that the CBF conditions hold over the entire continuous state space. This chapter bridges the gap by *jointly learning* a neural barrier function and its associated safe controller with Lipschitz-based verification guarantees for deterministic systems, where Lipschitz bounds on the barrier and its gradient suffice to certify safety. The extension to stochastic systems governed by SDEs is presented in Chapter 5.

3.2 Contributions of this Chapter

This chapter makes the following contributions:

- We propose a training framework to synthesize provably correct Control Barrier Functions (CBFs) parameterized as neural networks for continuous-time systems, eliminating any need for post hoc verification.
- Our methodology establishes completeness guarantees by deriving a validity condition, ensuring efficacy across the entire state space with only a finite number of data points. We train the network by enforcing Lipschitz bounds on the neural network and its Jacobian terms.
- We evaluate our approach using two case studies, namely, the inverted pendulum system and the obstacle avoidance of an autonomous driving system. We show that our training framework successfully constructs a verified NCBF to differentiate safe and unsafe regions.

The results presented in this chapter are based on [165, 197]. Code for this chapter is available online[1].

3.3 Problem Formulation

In this section, we begin by formally defining the problem of synthesis of CBF and highlighting the challenges associated with its direct solution. Subsequently, we propose a reformulation of the problem and derive the conditions, that provide formal guarantees on the correctness of the solution over the entire state set despite using finitely many data samples.

To leverage the universal approximation property of neural networks, we represent the Control Barrier Function with a feed-forward neural network $B_\theta : \mathcal{X} \to \mathbb{R}$, i.e., the network maps every state $x \in \mathcal{X}$ to a real scalar barrier value, where θ signifies the trainable parameters of the neural network; this representation is termed the Neural CBF (NCBF). Due to the absence of an NCBF, we lack access to the safe set $\mathcal{C}$. Hence, we start with *initial* safe and unsafe sets $\mathcal{X}_s \subseteq \mathcal{C}$ and $\mathcal{X}_u \subseteq \mathcal{X} - \mathcal{C}$, respectively. These are fixed, time-invariant subsets of the state space specified by the designer from prior knowledge of the system (e.g., operating limits or states whose safety is evident from physics); they are not time-varying regions, and they serve only to label training data. The learned certificate's zero-superlevel set $\widehat{\mathcal{C}} = \{x \in \mathcal{X} : B_\theta(x) \geq 0\}$ then estimates the true safe set $\mathcal{C}$, which may be considerably larger than $\mathcal{X}_s$. We require that any trajectory starting in $\mathcal{X}_s$ never enters $\mathcal{X}_u$. We next formulate the problem.

Objective 3.1. Given a continuous-time control system defined as (2.1), state set $\mathcal{X}$, initial safe and unsafe sets $\mathcal{X}_s$ and $\mathcal{X}_u$, respectively, the objective is to devise an algorithm to synthesize NCBF $B_\theta(x)$ using a Lipschitz continuous controller u such that

$$\begin{aligned} &B_\theta(x) \geq 0, \forall x \in \mathcal{X}_s, \\ &B_\theta(x) < 0, \forall x \in \mathcal{X}_u, \\ &\frac{\partial B_\theta(x)}{\partial x}(f(x) + g(x)u(x)) + \kappa\left(B_\theta(x)\right) \geq 0, \forall x \in \mathcal{X}. \end{aligned} \tag{3.1}$$

In order to enforce conditions (3.1) in Problem 3.1, we first cast our problem as

[1] https://github.com/tayalmanan28/Stochastic-NCBF

the following robust optimization problem (ROP):

$$\text{ROP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & \max\left(q_k(x)\right) \leq \psi, k \in \{1,2,3\} \\ & \forall x \in \mathcal{X}, \psi \in \mathbb{R}, \end{cases} \tag{3.2}$$

where

$$\begin{aligned} q_1(x) &= \left(-B_\theta(x)\right) \mathbb{1}_{\mathcal{X}_s}, \\ q_2(x) &= \left(B_\theta(x) + \delta\right) \mathbb{1}_{\mathcal{X}_u}, \\ q_3(x) &= -\frac{\partial B_\theta(x)}{\partial x}(f(x) + g(x)u(x)) - \kappa\left(B_\theta(x)\right), \end{aligned} \tag{3.3}$$

where δ is a small positive value to ensure the strict inequality. If the optimal solution of the ROP $(\psi^*_{\text{ROP}}) \leq 0$, then the conditions (3.1) are satisfied and the NCBF is valid.

However, the proposed ROP in (3.2) has infinitely many constraints since the state of the system is a continuous set. This motivates us to employ data-driven approaches and the scenario optimization program of ROP. Given a covering radius $\bar{\epsilon} > 0$, suppose we sample N data points $x_i \in \mathcal{X}, i \in \{1, \ldots, N\}$ forming an $\bar{\epsilon}$-*cover* of the state set: every $x \in \mathcal{X}$ satisfies $\|x - x_i\| \leq \bar{\epsilon}$ for at least one sample x_i. Instead of solving the ROP in (3.2), we employ the following scenario optimization problem (SOP):

$$\text{SOP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & q_1\left(x_i\right) \leq \psi, \forall x_i \in \mathcal{S}, \\ & q_2\left(x_i\right) \leq \psi, \forall x_i \in \mathcal{U}, \\ & q_3\left(x_i\right) \leq \psi, \forall x_i \in \mathcal{D}, \\ & \psi \in \mathbb{R}, i \in \{1, \ldots, N\}, \end{cases} \tag{3.4}$$

where $q_k(x), k \in \{1,2,3\}$ are defined as in (3.3). The data sets $\mathcal{S}, \mathcal{U}$ and $\mathcal{D}$ corresponding to points sampled from the initial safe set $\mathcal{X}_s \subseteq \mathcal{C}$, initial unsafe set $\mathcal{X}_u \subseteq \mathcal{X} - \mathcal{C}$, and state set $\mathcal{X}$, respectively.

Given the finite number of data samples x_i, and considering SOP as a linear program in relation to the decision variable ψ, it becomes feasible to find a solution for the SOP. Let us denote the optimal solution of the SOP as ψ^*. We now derive conditions under which the NCBF $B_\theta(x)$ satisfies conditions (3.1). The following theorem shows that solving (3.2) can be achieved by finding a solution to (3.4).

Assumption 3.1 (Lipschitz Continuity)**.** *The functions $q_k(x)$, $k \in \{1,2,3\}$, in (3.3) are globally Lipschitz continuous on $\mathcal{X}$ with finite constants L_k, i.e.,*

$|q_k(x) - q_k(x')| \leq L_k \|x - x'\|$ *for all* $x, x' \in \mathcal{X}$*; let* $L_{\max} := \max_k L_k$.

Two clarifications regarding Assumption 3.1 and the system model are in order. First, although *local* Lipschitz continuity of the vector field would suffice for well-posedness of the trajectories, Theorem 3.3.1 requires a uniform constant on $\mathcal{X}$ in order to propagate sample-level satisfaction across the entire $\bar{\epsilon}$-cover. Since $\mathcal{X}$ is compact throughout this chapter, this is equivalent to local Lipschitz continuity plus continuity on $\mathcal{X}$; more importantly, the required bounds (L_B, L_{dB}) are not merely postulated but are *actively enforced during training* through the LMI constraints of Section 3.4, so Assumption 3.1 holds by construction. Second, the formulation assumes knowledge of the drift and input maps f and g, which is needed to evaluate q_3 and to synthesize the QP-based controller; we discuss this restriction in the chapter summary and relax it in Chapter 7, where the controller is synthesized from learned dynamics.

Theorem 3.3.1. *Consider a continuous-time control system (2.1), and initial safe and unsafe sets* $\mathcal{X}_s \subseteq \mathcal{X}$ *and* $\mathcal{X}_u \subseteq \mathcal{X}$*, respectively. Let* $B_\theta(x)$ *be the neural network-based CBF with trainable parameters* θ*. For the SOP (3.4) constructed by utilizing* N *samples satisfying Assumption 3.1, let* ψ^* *be the optimal value. Then* $B_\theta(x)$ *is a valid* CBF*, i.e., it satisfies conditions (3.1), if the following condition holds:*

$$L_{\max}\bar{\epsilon} + \psi^* \leq 0, \tag{3.5}$$

where $L_{\max}$ *is defined in Assumption 3.1.*

Proof. For any $x \in \mathcal{X}$ and any $k \in \{1, 2, 3\}$, let x_i be a sample with $\|x - x_i\| \leq \bar{\epsilon}$. Then:

$$q_k(x) = q_k(x) - q_k(x_i) + q_k(x_i) \leq L_k \|x - x_i\| + \psi^* \leq L_k\bar{\epsilon} + \psi^* \leq L_{\max}\bar{\epsilon} + \psi^* \leq 0,$$

where the first inequality uses Assumption 3.1 and the second uses optimality ψ^* of the SOP (3.4). Hence, if condition (3.5) holds, then $q_k(x) \leq 0$ for all $x \in \mathcal{X}$ and all k, i.e., the $B_\theta(x)$ is a valid CBF satisfying conditions (3.1). □

Hence, the original Problem 3.1 can be efficiently addressed by solving the problem reformulated as follows.

Objective 3.2. Given a continuous-time control system defined as (2.1) and the data sets $\mathcal{S}$,$\mathcal{U}$ and $\mathcal{D}$, the objective is to devise an algorithm to synthesize NCBF $B_\theta(x)$ and consequently, NCBF $-$ QP based controller u, such that they satisfy the conditions required in SOP (3.4) (functions $q_k(x) \leq \psi^*, k \in \{1, 2, 3\}$

in equation (3.3)) and ψ^* satisfies condition (3.5).

Remark 3.3.2 (Relation Between Theorem 3.3.1 and Objective 3.2). Theorem 3.3.1 is a *verification* result: for a *fixed* controller $u(\cdot)$—and hence fixed functions q_k—it certifies that a candidate B_θ satisfies conditions (3.1) over all of $\mathcal{X}$ whenever (3.5) holds. Objective 3.2, by contrast, is a *synthesis* statement that additionally searches for the controller: the training procedure alternates between synthesizing u_i via the NCBF-QP for the current barrier B_θ (thereby changing q_3) and updating (B_θ, ψ) until the SOP conditions hold at every sample and the validity condition (3.5) is met. Once this procedure converges, the pair (B_θ, u) is fixed, and Theorem 3.3.1 applies verbatim to certify safety of the closed-loop system.

3.4 Method

In this section, we propose an algorithmic approach to solve the problem formulated in Section 3.3. The structure of this section is as follows. We first present the method to synthesize NCBF to solve Problem 3.2 and then demonstrate the training process.

3.4.1 Synthesis of NCBF

Following the problem formulation in Section 3.3, we now describe the construction of suitable loss functions for the training NCBF $B_\theta(x)$ such that its minimization leads to the solution of Problem 3.2. As described in the previous section, NCBF is a feed-forward neural network, with trainable weight parameters θ. To address the aforementioned problem, it is imperative to compute $\frac{\partial B_\theta(x)}{\partial x}$. This necessitates the selection of a neural network with a smooth activation function, thereby facilitating the derivation of smooth Jacobian values upon differentiation. Interested readers are directed to [103], which analytically computes the Jacobian using the chain rule.

Now, let us consider the following loss functions satisfying the conditions required by SOP (3.4) over the training data sets $\mathcal{S}, \mathcal{U}, \mathcal{D}$ as follows:

$$\begin{aligned}
\mathcal{L}_1(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{S}} \max\left(0, q_1(x_i) - \psi\right), \\
\mathcal{L}_2(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{U}} \max\left(0, q_2(x_i) - \psi\right), \\
\mathcal{L}_3(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{D}} \max\left(0, q_3(x_i) - \psi\right)
\end{aligned} \tag{3.6}$$

with $\mathcal{L}_1$ representing the loss for safe states, $\mathcal{L}_2$ representing the loss for unsafe states and $\mathcal{L}_3$ representing the loss for lie derivative conditions over the entire state set, respectively. Using the terms defined above, the loss function is given by

$$\mathcal{L}_\theta(\theta) = \mathcal{L}_1 + \lambda_1 \mathcal{L}_2 + \lambda_2 \mathcal{L}_3, \tag{3.7}$$

with $\lambda_1, \lambda_2 \in \mathbb{R}_+$weighting the importance of the individual loss terms.

To ensure the fulfillment of the assumptions in Theorem 3.3.1, it is imperative to verify the Lipschitz boundedness of the functions $q_k(x)$, where $k \in \{1, 2, 3\}$. This necessitates the Lipschitz boundedness of $B_\theta(x)$ and $\frac{\partial B_\theta(x)}{\partial x}$, with corresponding Lipschitz bounds denoted as L_B and L_{dB}, respectively. To train neural networks with Lipschitz bounds, we have the following lemma:

Lemma 3.1 ([117]). *Suppose f_θ is a l-layered feed-forward neural network with θ as a trainable parameter, then a certificate for L-Lipschitz continuity of the neural network is given by the semi-definite constraint $M(\theta, \Lambda) :=$*

$$\begin{bmatrix} A \\ B \end{bmatrix}^T \begin{bmatrix} 2\alpha\beta\Lambda & -(\alpha+\beta)\Lambda \\ -(\alpha+\beta)\Lambda & 2\Lambda \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} + \begin{bmatrix} L^2\boldsymbol{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\theta_l^T \\ 0 & 0 & -\theta_l & \boldsymbol{I} \end{bmatrix} \succeq 0,$$

where

$$A = \begin{bmatrix} \theta_0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \theta_{l-1} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & I \end{bmatrix},$$

$(\theta_0, \dots \theta_l)$ are the weights of the neural network, $\Lambda \in \mathbb{D}_+^n$, $i = 1, \dots, l$ and α and β are the minimum and maximum slopes of the activation functions, respectively.

Remark 3.4.1. For a single-layer case, the matrix M reduces to:

$$M(\theta, \Lambda) = \begin{bmatrix} L^2\boldsymbol{I} + 2\alpha\beta\theta_0^T\Lambda\theta_0 & -(\alpha+\beta)\theta_0^T\Lambda & 0 \\ -(\alpha+\beta)\Lambda\theta_0 & 2\Lambda & -\theta_1 \\ 0 & -\theta_1 & \boldsymbol{I} \end{bmatrix} \succeq 0.$$

The lemma above addresses the certification of the L-Lipschitz bound for a neural network. However, our scenario necessitates ensuring not only the Lipschitz boundedness of the NCBF $\tilde{B}$, but also of $\frac{\partial B_\theta(x)}{\partial x}$. Therefore, we must explore the relationship between the network weights and the semi-definite matrix M to guarantee the bound-

edness of the aforementioned terms. To address this issue, we introduce the following theorem.

Theorem 3.4.2. *Consider a 1-layered feedforward neural network f_θ, with output dimension 1×1, where θ represents the trainable weight parameters. Let y denote the output of the neural network, x denote the input of the neural network, θ_i, $i \in 0, 1$, denote the weight parameters of each layer with $\theta = \{\theta_0, \theta_1\}$ and ϕ denote the activation. The certificate for L-Lipschitz continuity of the derivative of the neural network $(\frac{\partial y}{\partial x})$ is given by $M_{\hat{\phi}}(\hat{\theta}, \Lambda) \succeq 0$, where $\hat{\phi} = \phi'$ and $\hat{\theta} = (\theta_0, \hat{\theta}_1)$, $\hat{\theta}_1$ is defined as:*

$$\hat{\theta}_1 = \theta_0^T \mathsf{diag}(\theta_1). \tag{3.8}$$

Proof. The proof can be found in Appendix B.1.2. □

We address a constrained optimization problem aiming to minimize loss $\mathcal{L}(f_\theta)$ subject to $\boldsymbol{M}_j(\theta, \Lambda) \succeq 0$ for $j = 1, 2$, where f_θ is a given neural network. By employing a log-det barrier function, we convert this into an unconstrained optimization problem:

$$\min_{\theta, \Lambda} \mathcal{L}(f_\theta) + \mathcal{L}_M(\theta, \Lambda),$$

where $\mathcal{L}_M(\theta, \Lambda) = -\sum_{j=0}^{q} \rho_j \log\det(\boldsymbol{M}_j(\theta, \Lambda))$ and $\rho_j > 0$ are barrier parameters. Ensuring that the loss function $\mathcal{L}_M(\theta, \Lambda) \leq 0$, guarantees that the linear matrix inequalities $\boldsymbol{M}_j(\theta, \Lambda) \succeq 0, j = 1, 2$ hold true. Let us consider the loss functions characterizing the satisfaction of Lipschitz bound as

$$\mathcal{L}_M(\theta, \Lambda, \hat{\Lambda}) = -c_{l_1} \log\det(M_1(\theta, \Lambda)) - c_{l_2} \log\det(M_2(\hat{\theta}, \hat{\Lambda})), \tag{3.9}$$

where c_{l_1}, c_{l_2} are positive weight coefficients for the sub-loss functions, M_1, M_2 are the semi-definite matrices corresponding to the Lipschitz bounds L_B, L_{dB} respectively, $\Lambda, \hat{\Lambda}$ are trainable parameters and $\theta, \hat{\theta}$ are the weights mentioned in Theorem 3.4.2.

Finally, let us consider the following loss function to satisfy validity condition (3.5):

$$\mathcal{L}_v(\psi) = \max(0, L_{\max}\bar{\epsilon} + \psi), \tag{3.10}$$

where $L_{\max}$ is maximum of the Lipschitz constants of $q_k(x), k \in \{1, 2, 3\}$ in (3.3), or $L_{\max} = \max(L_B, L_B + L_{dB}L_x)$.

Algorithm 1 Learning Formally Verified Neural Control Barrier Functions

Require: Inputs: Data Sets: $\mathcal{S}, \mathcal{U}, \mathcal{D}$; Dynamics: f, g; Lipschitz Bounds: L_B, L_{dB}, L_x; Cover radius: $\bar{\epsilon}$; Initial network parameters $\theta^{(0)}$

Ensure: Outputs: Verified NCBF parameters θ^* and margin ψ^* satisfying (3.5); certified controller via NCBF-QP (3.11)

```
Initialise(θ, ψ, Λ, Λ̂)
x_i ← sample(S, U, D)
L_max ← max(L_B, L_B + L_dB L_x)
while L_θ > 0 or L_M ⋠ 0 or L_v > 0 do
    B̃ ← B_θ
    u_i ← NCBF − QP(B̃, f, g, x_i, ψ)        ▷ From eq. (3.11)
    L_θ ← (B̃, f, g, x_i, u_i, ψ)            ▷ From eq. (3.7)
    θ ← Learn(L_θ, θ)
    L_M ← (θ, Λ, Λ̂)                         ▷ From eq. (3.9)
    θ, Λ, Λ̂ ← Learn(L_M)
    L_v ← (L_max, ψ)                        ▷ From eq. (3.10)
    ψ ← Learn(L_v)
end while
return θ* ← θ, ψ* ← ψ
```

3.4.2 Training with Safety Guarantee

The overall training procedure is as follows. We first fix all the hyper-parameters required for the training, including $\bar{\epsilon}, \mathcal{L}_B, \mathcal{L}_{dB}, \lambda_1, \lambda_2, c_{l_1}, c_{l_2}$, and the maximum number of epochs considered. At each time step, the controller intakes the current state x and reference control input $u_{\text{ref}}(x)$. We emphasize that u_{ref} is *not* an output of the algorithm: it is any user-supplied nominal policy (e.g., an energy-shaping controller for the pendulum or a lane-keeping controller in the driving study below) that is passed to the QP at deployment time. The basic idea is to construct safe control input by filtering out unsafe actions based on the condition derived from learned NCBF. Specifically, the controller minimizes the modification on the control input u compared to u_{ref} such that the NCBF safety condition holds by solving NCBF $-$ QP defined as follows.

$$\begin{aligned} &\min_{u \in \mathcal{U}} \|u - u_{\text{ref}}(x)\| \\ &\text{s.t.} \frac{\partial B_\theta(x)}{\partial x}(\mathbf{f}(x_i) + \mathbf{g}(x_i)u) \geq -\kappa(B(x_i)). \end{aligned} \tag{3.11}$$

Finally, the control action u can be used in computing loss functions in the next iteration. The training data sets are randomly shuffled into several batches, and the loss is calculated for each batch at a time. Then, using a stochastic gradient descent algorithm, e.g., ADAM, the trainable parameters $\theta, \Lambda, \hat{\Lambda}$ and ψ are updated. This procedure is repeated until the network converges or if a predefined maximum episode

number is reached. For our proposed approach, all of the learning is done offline in a simulation environment. After the learning process converges, the NCBF-QP based controller u can then be deployed to the intended system. The overall algorithm is summarised in Algorithm 1

We next present the following theorem that provides formal guarantees of safety for the continuous-time control system by utilizing the trained NCBF $B_\theta(x)$.

Theorem 3.4.3. *Consider a continuous-time control system* (2.1)*, and safe and unsafe sets* $X_s \subseteq X$ *and* $X_u \subseteq X$*, respectively. Let* $B_\theta(x)$ *be the trained* NCBF*, such that* $\mathcal{L}_\theta = 0$*,* $\mathcal{L}_M \leq 0$ *and* $\mathcal{L}_v = 0$*. Then, the control action* u *obtained by solving the* NCBF $-$ QP (3.11)*, makes the system safe.*

Remark 3.4.4 (Generalization Beyond Training Samples). Although training uses finitely many samples, the certificate is *not* restricted to them. The combination of the $\bar{\epsilon}$-cover of $\mathcal{X}$ and the enforced Lipschitz bounds ensures that satisfying $q_k(x_i) \leq \psi^*$ at every sample implies $q_k(x) \leq \psi^* + L_{\max}\bar{\epsilon}$ at *every* $x \in \mathcal{X}$; the validity condition (3.5) absorbs precisely this worst-case propagation error. Approximation errors of the network are therefore accounted for through the margin ψ^*: the trained NCBF generalizes to unseen states because its Lipschitz bounds hold globally by construction (Lemma 3.1 and Theorem 3.4.2), rather than depending on data density. States far from the sampled grid are covered conservatively: the certificate degrades gracefully through the $L_{\max}\bar{\epsilon}$ slack instead of failing unpredictably, which is also why the resulting guarantee is stronger than purely sample-based (e.g., conformal) certificates—at the price of conservativeness.

3.5 Simulations

In this section, we assess the efficacy of our proposed methodology through two distinct case studies: the inverted pendulum model and the obstacle avoidance of an autonomous mobile robot [14]. Both case studies are conducted on a computing platform equipped with an Intel i7-12700H CPU, 32GB RAM, and NVIDIA GeForce RTX 3090 GPU. For the class $\mathcal{K}$ function in the CBF inequality (2.10), we chose $\kappa(B) = \gamma B$, where $\gamma = 1$.

3.5.1 Inverted Pendulum

We consider a continuous-time inverted pendulum dynamics given as follows:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u, \tag{3.12}$$

where $\theta \in \mathbb{R}$ denotes the angle, $\dot{\theta} \in \mathbb{R}$ denotes the angular velocity, $u \in \mathbb{R}$ denotes the controller, m denotes the mass and l denotes the length of the pendulum. We let the mass $m = 1$kg and length $l = 10$m. The inverted pendulum operate in a state space given as $\mathcal{X} = \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]^2$. The system is required to stay in a limited safe stable region. The initial safe region is given as $\mathcal{X}_s = \left[-\frac{\pi}{15}, \frac{\pi}{15}\right]^2$ and the unsafe region is given as $\mathcal{X}_u = \mathcal{X} \setminus \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]^2$.

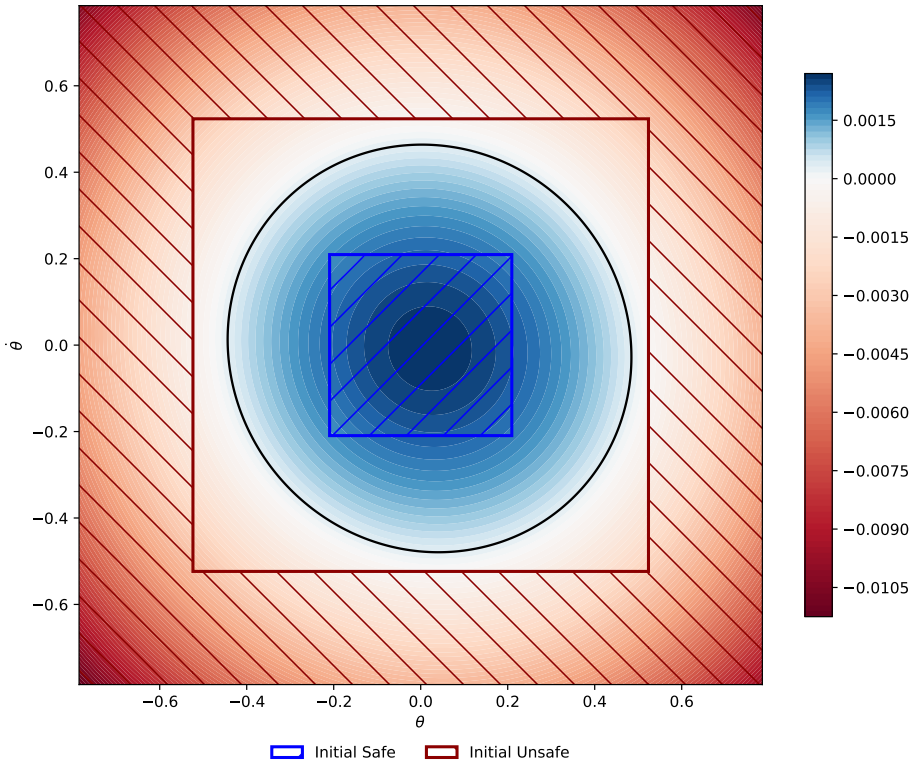


Figure 3.1: Plot of $B_\theta(x)$ over X. The horizontal and vertical axes are the pendulum angle θ [rad] and angular velocity $\dot{\theta}$ [rad/s], respectively. Blue and red regions denote the safe region ($B \geq 0$) and the unsafe region ($B < 0$), respectively. The initial safety region boundary and unsafe region boundary is denoted by their respective regions marked with coloured oblique lines. The black line denotes the zero-level contour of the barrier function. We note that the framework does not require square or box-constrained safe/unsafe sets; such sets are used here purely for ease of specification, and arbitrary (e.g., polytopic or nonconvex) sets can be handled identically since the method relies only on sampling and Lipschitz bounds.

We train the NCBF, assuming knowledge of the model. The NCBF B consists of

one hidden layer of 20 neurons, with Softplus activation function ($log(1 + exp(x))$.

We set the training hyper-parameters to $\bar{\epsilon} = 0.00016$, $L_B = 0.01$ and $L_{dB} = 0.4$ yielding $L_{\max} = 0.41$. We perform the training to simultaneously minimize the loss functions $\mathcal{L}_\theta, \mathcal{L}_M$, and $\mathcal{L}_v$. The training algorithm then converges to obtain the NCBF $B_\theta(x)$ with $\psi^* = -0.00042$. Thus, using Theorem 3.3.1, we can verify that the NCBF thus obtained is valid, thus ensuring safety.

Visualizations of the trained NCBF are presented in Fig. 3.1. These visualizations demonstrate the successful separation of the initial safety region boundary from the unsafe region boundary.

3.5.2 Obstacle Avoidance

We consider an autonomous mobile robot navigating on a road following the dynamics [50] given below

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{3.13}$$

where $[x_1, x_2, \psi]^T \in \mathcal{X} \subseteq \mathbb{R}^3$ is the state consisting of the location (x_1, x_2) of the robot and its orientation ψ, with v representing the robot's speed and u controlling its orientation. We set the speed $v = 1$.

The mobile robot is required to stay on the road while avoiding pedestrians sharing the field of activities. The unicycle operates in a state space given as $\mathcal{X} = [-2, 2]^3$. The initial safe region is given as $\mathcal{X}_s = \mathcal{X} \backslash [-1.5, 1.5]^2 \times [-2, 2]$ and the unsafe region is given as $\mathcal{X}_u = [-0.2, 0.2]^2 \times [-2, 2]$.

We train the NCBF, assuming knowledge of the model with the same architecture as before (one hidden layer of 20 neurons, with Softplus activation function ($log(1 + exp(x))$). The training hyper-parameters are set to $\bar{\epsilon} = 0.01$, $L_B = 1$ and $L_{dB} = 1$ resulting in $L_{\max} = 2$. We perform the training to simultaneously minimize the loss functions $\mathcal{L}_\theta, \mathcal{L}_M$, and $\mathcal{L}_v$. The training algorithm then converges to obtain the NCBF $B_\theta(x)$ along with $\psi^* = -0.04002$.

We validate our safe control in the CARLA simulation environment, in which the vehicle maneuvers its orientation to conduct obstacle avoidance. If the control policy guides the vehicle to another lane with no obstacle ahead, then it switches to a baseline autonomous driving algorithm. The trajectory of the vehicle is shown in Fig. 3.2. We show the trajectory of the proposed NCBF-based safe control in different reaction distances, namely, 6, 8, and 12 meters, respectively. All three trajectories

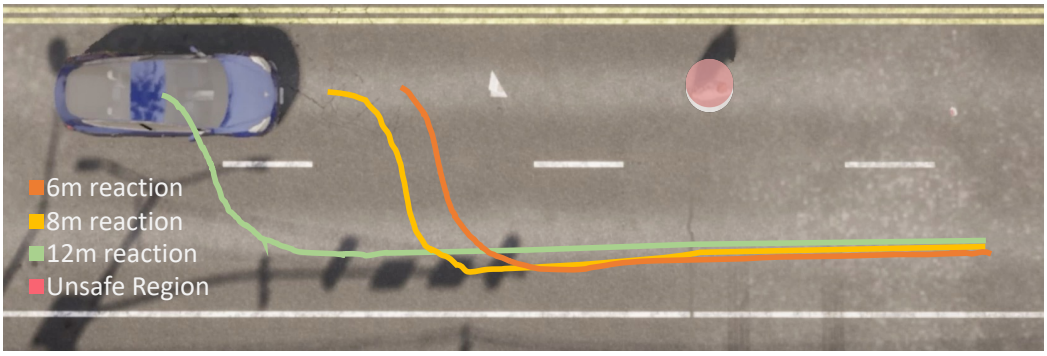


Figure 3.2: Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed NCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

of the vehicle under control show our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

We next compare our proposed method with NCBF trained with baseline proposed in [195] assuming given function $B(x)$. Among existing NCBF training schemes, this baseline is the most closely related approach that also provides training-time satisfaction guarantees for a neural barrier; we therefore run both methods with identical network architectures, sampling budgets, and hyperparameters, so that the comparison isolates the difference between the two training schemes rather than implementation details. The synthesized NCBF are visualized in Fig. 3.3. We observe that the synthesized NCBF B successfully separates the safe and the unsafe region, since both zero-level sets (in yellow) do not overlap with the unsafe region in red color. We also found that the proposed method ensures a larger subset compared with the baseline. The baseline ensures that 11.8% of the state space is safe, while the proposed method ensures that 77.85% of the state space is safe.

3.5.3 Cart-Pole

To evaluate the method on a more strongly coupled system, we additionally consider the four-dimensional cart-pole state $x = (p, v, \theta, \omega)$, where p and v are the cart position and velocity and θ and ω are the pole angle and angular velocity. The dynamics are

$$\dot{x} = \begin{bmatrix} v \\ a_0(x) \\ \omega \\ b_0(x) \end{bmatrix} + \begin{bmatrix} 0 \\ a_1(x) \\ 0 \\ b_1(x) \end{bmatrix} u, \tag{3.14}$$

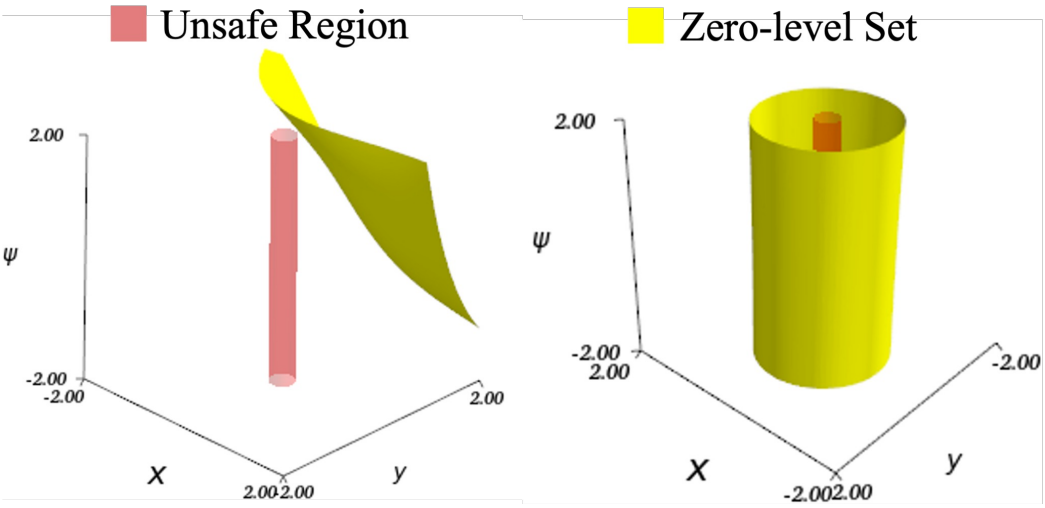


Figure 3.3: The left and right figures show the unsafe region and the zero-level sets of the NCBF B trained by baseline and the proposed method, respectively. Both zero-level sets (in yellow) do not overlap with the unsafe region in red color. The NCBF trained by the baseline ensures a guaranteed safe subset over the state space with the ratio 11.8% only, whereas the proposed method ensures a guaranteed safe subset with a ratio up to 77.85%.

where

$$d = l\left(\frac{4}{3} - \frac{m\cos^2\theta}{M+m}\right), \qquad t_0 = \frac{ml\omega^2\sin\theta}{M+m}, \tag{3.15}$$

$$b_0(x) = \frac{g\sin\theta - \cos\theta\, t_0}{d}, \qquad b_1(x) = -\frac{\cos\theta}{(M+m)d}, \tag{3.16}$$

$$a_0(x) = t_0 - \frac{mlb_0(x)\cos\theta}{M+m}, \qquad a_1(x) = \frac{1}{M+m} - \frac{mlb_1(x)\cos\theta}{M+m}. \tag{3.17}$$

We use $M = 1$, $m = 0.1$, $l = 0.5$, and $g = 9.81$. The state domain and initial set are

$$\mathcal{X} = [-1, 1] \times [-0.5, 0.5] \times [-0.4, 0.4] \times [-1, 1], \tag{3.18}$$

$$\mathcal{X}_s = [0.1, 0.2] \times [-0.1, 0.1] \times [-0.05, 0.05] \times [-0.1, 0.1], \tag{3.19}$$

and the unsafe set for this experiment is $\mathcal{X}_u = \{x \in \mathcal{X} : p < 0\}$.

The NCBF uses one hidden layer with 64 neurons and Softplus activation. We construct an anisotropic cover with $(41, 21, 41, 31)$ centres along (p, v, θ, ω) and half-cell widths

$$\left(\frac{1}{41}, \frac{1}{42}, \frac{0.4}{41}, \frac{1}{31}\right), \tag{3.20}$$

giving $\bar{\epsilon} = 0.0479326777$ and 1,094,331 sampled centres. The listed bounds are $L_B = 20$, $L_{dB} = 1$, $L_{d^2B} = 2$, and $L_x = 20$. The training converged to an NCBF with $\psi^* = -0.121$, and the validation conditions were satisfied for the reported cart-pole experiment. The need to evaluate more than one million centres for this four-dimensional

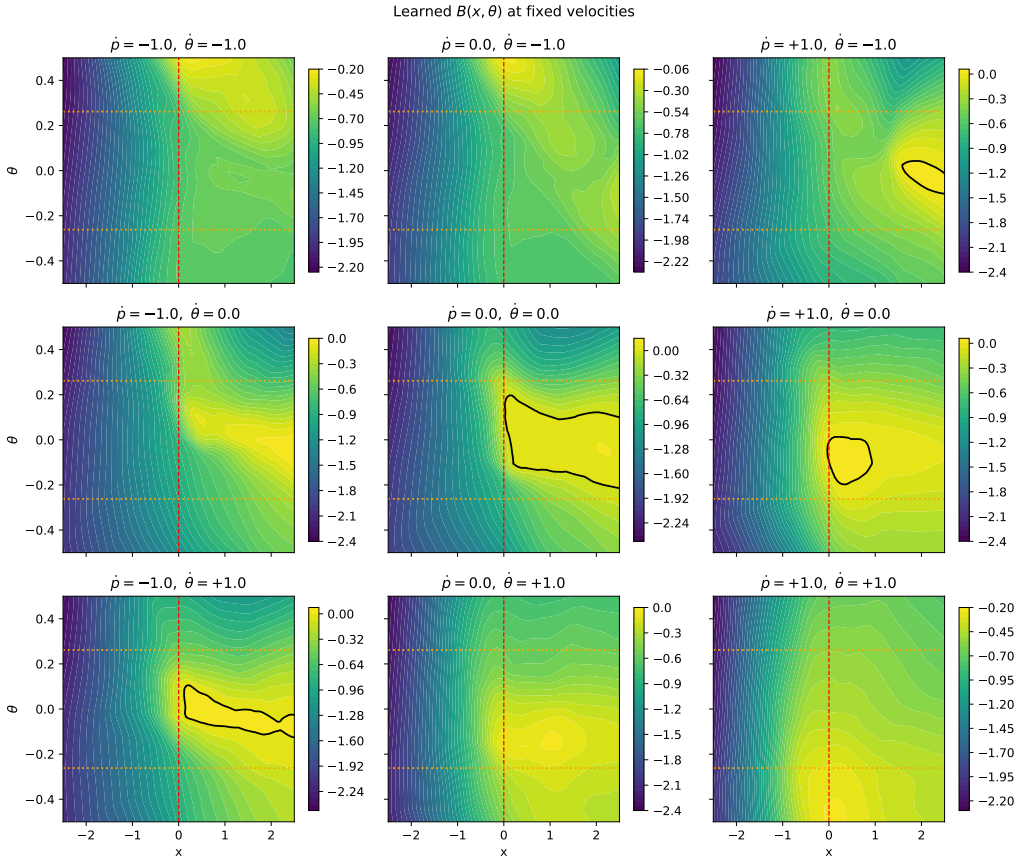


Figure 3.4: Learned cart-pole NCBF evaluated over the velocity grid.

example illustrates the computational scaling of the dense-cover construction relative to the lower-dimensional examples. Figure 3.4 visualizes the learned cart-pole certificate over the velocity grid.

3.6 Chapter Summary

This chapter presented Neural Control Barrier Functions (NCBFs) with formal Lipschitz-based verification for deterministic dynamical systems governed by ODEs. By combining Lipschitz-constrained neural networks with robust scenario optimization, we developed a training procedure that simultaneously learns a neural barrier function and a safe controller with formally verified safety guarantees. Experiments on benchmark systems and in the CARLA autonomous driving simulator demonstrated that the proposed method certifies significantly larger safe sets compared to baselines.

The NCBF framework establishes that *learning-based* safety certificates can provide formal guarantees comparable to, and in some cases stronger than, their handcrafted counterparts. The cover-based construction is deliberately the most conservative verification point in this thesis: it trades scalability for a deterministic certificate

over the covered continuous state space. Its dimensional cost motivates the subsequent progression to conformal calibration in Chapter 4, verification-in-the-loop synthesis in Chapter 5, offline value-guided barriers in Chapter 7, physics-informed safety–performance co-optimization in Chapter 8, and decentralized multi-agent learning in Chapter 9. Two key limitations therefore emerge. First, the Lipschitz-based verification relies on *global* bounds on the network and its derivatives; as network size or state dimension increases, these bounds become increasingly conservative, limiting scalability to complex systems. Second, the framework does not provide tools for verifying approximation errors that arise when the barrier function is evaluated at states not seen during training, a gap that grows more pronounced as the data coverage becomes sparse in high-dimensional spaces.

A further limitation—flagged explicitly here since f and g are inputs to both the training procedure (Algorithm 1) and the deployment-time QP (3.11)—is the reliance on exact knowledge of the system dynamics. The certificate of Theorem 3.3.1 characterizes neither unmodeled dynamics nor external disturbances: if the realized vector field deviates from the nominal (f, g) used at training time, the guarantee degrades by an amount proportional to the mismatch. A principled robustness treatment could follow standard robust-CBF constructions [191, 75], which tighten the barrier condition by a term proportional to $\|\nabla B\|\bar{d}$ for disturbance bound $\bar{d}$, and subsequent chapters progressively address these concerns: Chapter 5 extends the synthesis-and-verification framework to stochastic systems with diffusion terms, while Chapter 7 removes the reliance on exactly known dynamics during controller synthesis. We view a full robustness analysis for modeling errors in the present Lipschitz framework as an important direction for future work.

In the next chapter, we address these limitations by introducing *conformal prediction* as an alternative verification mechanism. Conformal prediction provides finite-sample, distribution-free guarantees that do not depend on global Lipschitz constants, enabling less conservative safe sets and better scalability to higher-dimensional systems.

Chapter 4

Conformal Neural CBFs

4.1 Related Works

Chapter 3 demonstrated that Neural CBFs can be equipped with formal safety guarantees in deterministic settings through Lipschitz-based verification. While effective, this approach has inherent limitations: global Lipschitz bounds grow with network depth and width, leading to increasingly conservative safety margins as the system dimensionality increases. This motivates the search for verification tools that provide formal guarantees without relying on worst-case Lipschitz constants.

Verification of Neural Certificates. Verifying that a learned barrier function satisfies its defining conditions over the entire continuous state space remains a central challenge. SMT-based methods [1] and mixed-integer programming [171] provide exact verification but are limited to small networks and low-dimensional systems. Interval-bound propagation and linear relaxations [60] scale better but produce loose bounds. The Lipschitz approach used in Chapter 3 [165] strikes a balance but requires expensive LMI constraints and yields conservative certificates in high dimensions.

Conformal Prediction. Conformal prediction [175, 138] is a distribution-free statistical framework that constructs prediction sets with guaranteed marginal coverage from finitely many calibration samples. Split conformal prediction [115] is computationally efficient and requires only exchangeability of the calibration data. Recent works have applied conformal prediction to robotics for uncertainty quantification in trajectory prediction [97], motion planning [49], and perception [10]. However, its use for *verifying* learned safety certificates—certifying that a neural barrier function satisfies its defining conditions with high probability over the entire state space—has not been explored prior to this work.

Gap Addressed. This chapter introduces conformal prediction as a scalable verification tool for Neural CBFs in deterministic systems. Unlike Lipschitz-based methods, our approach provides finite-sample probabilistic guarantees that do not require global Lipschitz bounds, enabling less conservative safe sets and better scalability to high-dimensional systems.

4.2 Contributions of this Chapter

This chapter makes the following contributions:

1. We propose *CP-NCBF*, a framework that integrates split conformal prediction into the NCBF synthesis pipeline to provide *finite-sample, distribution-free probabilistic safety guarantees* without requiring global Lipschitz constants.
2. We develop an iterative training procedure in which the conformal score $\hat{q}$ is used to update the safety margin ψ, which in turn refines the NCBF. This process converges when $\hat{q} = 0$, certifying that the learned barrier satisfies its conditions with the desired probability level.
3. We demonstrate that CP-NCBF produces *less conservative* safe sets compared to Lipschitz-verified NCBFs [165] while maintaining equivalent or superior empirical safety rates, with improved sample efficiency.
4. We validate the framework on autonomous driving and aerial vehicle geo-fencing tasks, including a high-dimensional quadruped navigation setting, demonstrating scalability beyond what Lipschitz-based verification can achieve.

The results presented in this chapter are based on [158].

4.3 Problem Formulation

In this section, we formally define the problem of Control Barrier Function (CBF) synthesis and outline the challenges inherent to its direct formulation, particularly those stemming from the need to verify correctness over the entire continuous state space. To overcome these challenges, we propose a reformulation that admits a probabilistic interpretation, enabling formal guarantees on the validity of the solution across the entire state space despite reliance on finitely many data samples.

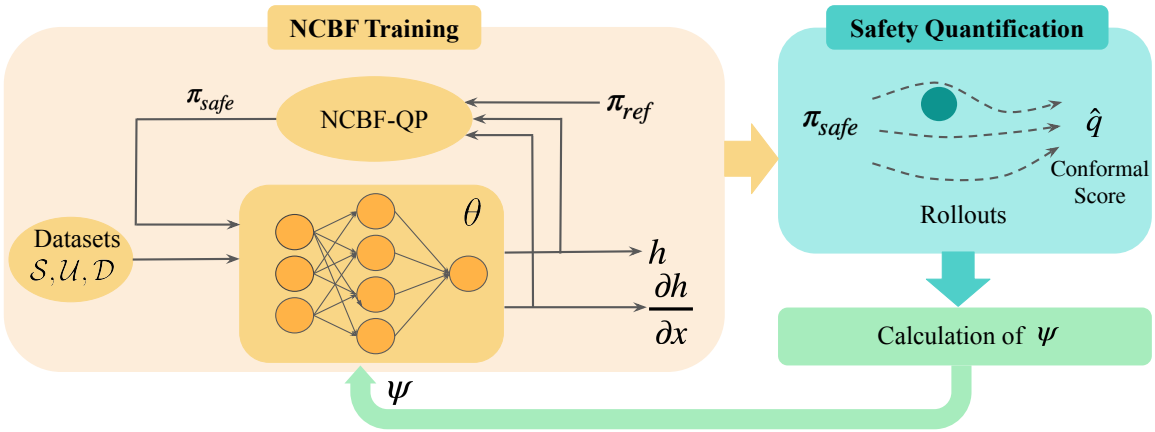


Figure 4.1: **Overview of the proposed approach**: The proposed methodology consists of **three main steps**. In the **first step**, the NCBF, denoted by B_θ, is trained using an initial safety margin parameter $\psi = 0$. **In the second step**, the conformal score $\hat{q}$ is computed based on the policy induced by the learned NCBF. **In the third step**, this conformal score is then used to update the safety margin ψ, which is subsequently employed to retrain the NCBF. This iterative process is repeated until the conformal score converges to zero, thereby ensuring the desired probabilistic safety guarantee.

Using the universal approximation property of neural networks, we introduce a Neural CBF (NCBF) parameterized by a deep neural network (DNN), $B_\theta(x)$, trained via a semi-supervised learning scheme. Throughout this chapter, we consider continuous-time control-affine systems of the form

$$\dot{x} = f(x) + g(x)u, \tag{4.1}$$

with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, control $u \in \mathcal{U} \subseteq \mathbb{R}^m$, drift dynamics $f : \mathcal{X} \to \mathbb{R}^n$, and input map $g : \mathcal{X} \to \mathbb{R}^{n\times m}$. In the absence of a known CBF, and consequently the true safe set $\mathcal{C}$, we begin with approximate sets: an initial safe set $\mathcal{X}_s \subseteq \mathcal{C}$ and an initial unsafe set $\mathcal{X}_u \subseteq \mathcal{X} \setminus \mathcal{C}$ (fixed, designer-specified subsets used to label training data), such that any trajectory originating in $\mathcal{X}_s$ does not enter $\mathcal{X}_u$. Based on this setup, we proceed to formally state the learning problem.

Objective 4.1. Given a continuous-time control system defined in Equation (4.1), state space $\mathcal{X}$, initial safe and unsafe sets $\mathcal{X}_s$ and $\mathcal{X}_u$, respectively, the objective is to devise an algorithm to synthesize NCBF $B_\theta(x)$ using a Lips-

chitz continuous controller u such that

$$\begin{aligned} B_\theta(x) &\geq 0, \forall x \in \mathcal{X}_s, \\ B_\theta(x) &< 0, \forall x \in \mathcal{X}_u, \\ \frac{\partial B_\theta}{\partial x}(f(x) + g(x)u(x)) + \kappa\left(B_\theta\right) &\geq 0, \forall x \in \mathcal{X}. \end{aligned} \tag{4.2}$$

Instead of solving the infinite-dimensional robust program, we enforce its constraints on the sampled data through the following scenario optimization program (SOP):

$$\begin{aligned} \underset{\psi \in \mathbb{R}}{\text{minimize}} \quad & \psi \\ extsubjectto \quad & q_1(x_i) := -B_\theta(x_i) \leq \psi, && x_i \in \mathcal{S}, \\ & q_2(x_i) := B_\theta(x_i) \leq \psi, && x_i \in \mathcal{U}, \\ & q_3(x_i) := -\nabla B_\theta(x_i)\big(f(x_i) + g(x_i)u(x_i)\big) - \kappa(B_\theta(x_i)) \leq \psi, && x_i \in \mathcal{D}. \end{aligned} \tag{4.3}$$

Here, $\mathcal{S}$, $\mathcal{U}$, and $\mathcal{D}$ contain samples from the initial safe set, initial unsafe set, and state space, respectively. A small strict-separation margin may be included in q_2 in implementations.

Given the finite set of data samples x_i and the linear structure of the SOP with respect to the decision variable ψ, the optimization problem is feasible. We denote the optimal solution of the SOP, representing a uniform *safety margin*—a training-time slack on the CBF conditions that governs conservatism (distinct from robustness margins against model uncertainty or disturbances)—as ψ^*. Using this premise, we define the primary objective of the paper:

Objective 4.2. Given a continuous-time control-affine nonlinear dynamical system and the datasets $\mathcal{S}$, $\mathcal{U}$, and $\mathcal{D}$, our objective is to develop a framework for synthesizing an NCBF B_θ using a uniform safety margin ψ, such that it satisfies the SOP in (4.3) with a user-specified probabilistic confidence level. The resulting statement is a marginal, distribution-relative guarantee for a new exchangeable state, rather than deterministic set invariance over every state.

4.4 Conformal Neural CBF

In this section, we propose an algorithmic approach to solve the problem formulated in Section 3.3. The structure of this section is as follows. We first present the method

to synthesize an NCBF to realize Objective 4.2 and then demonstrate the training process.

4.4.1 Synthesis of NCBF

Following the problem formulation in Section 3.3, we now present the construction of loss functions for training the NCBF $B_\theta(x)$ such that their minimization facilitates the realization of Objective 4.2. As discussed previously, the NCBF is a feed-forward neural network with trainable weights θ.

Network architecture. We instantiate B_θ as a multilayer perceptron with Softplus activations; the per-task architectures are tabulated in Appendix C.4 (one hidden layer of 64 units for the ground vehicle; two hidden layers of 128 units for the aerial and quadruped tasks). We emphasize that CP-NCBF imposes *no architectural requirements*: the conformal calibration of Section 4.4 is performed *post hoc* on whatever network has been trained, so any smooth parameterization is admissible—including the single-hidden-layer networks used in Chapter 3. Deeper architectures are adopted here solely for expressive capacity in the higher-dimensional case studies ($n = 6$–8), where shallow networks underfit the barrier geometry; the guarantees themselves are architecture-agnostic.

To ensure conformity with the requirements of the SOP in (4.3), we define the following loss functions over the training datasets $\mathcal{S}$, $\mathcal{U}$, and $\mathcal{D}$:

$$\begin{aligned} \mathcal{L}_1(\theta) &= \frac{1}{N}\sum_{x_i\in\mathcal{S}} \max\left(0, q_1(x_i) - \psi^*\right), \quad \mathcal{L}_2(\theta) = \frac{1}{N}\sum_{x_i\in\mathcal{U}} \max\left(0, q_2(x_i) - \psi^*\right), \\ \mathcal{L}_3(\theta) &= \frac{1}{N}\sum_{x_i\in\mathcal{D}} \max\left(0, q_3(x_i) - \psi^*\right), \quad \mathcal{L}_\theta(\theta) = \mathcal{L}_1 + \lambda_1\mathcal{L}_2 + \lambda_2\mathcal{L}_3, \end{aligned} \tag{4.4}$$

where $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ penalize violations of the CBF condition in the safe, unsafe, and Lie derivative constraint regions, respectively. The overall loss $\mathcal{L}_\theta$ is a weighted combination of these components, with $\lambda_1, \lambda_2 \in \mathbb{R}_+$ denoting the relative importance of each term. This formulation enables the systematic training of the NCBF with the safety margin ψ^*.

4.4.2 Conformal Neural CBF

From Equation (4.4), it is evident that the choice of ψ plays a critical role in shaping the learned neural CBF. Specifically, setting $\psi > \psi^*$ can lead to safety violations,

thereby rendering the CBF incomplete, whereas choosing $\psi \ll \psi^*$ introduces unnecessary conservatism in the control policy. A fundamental challenge, therefore, lies in determining an appropriate value for ψ^*. Intuitively, ψ^* represents the worst-case safety violation across the entire state space, implying that accurate estimation of ψ^* requires computing a tight upper bound on the maximum violation over an infinite verification set. However, evaluating the system over infinitely many state-space points is computationally intractable.

To overcome this limitation, we propose a split conformal prediction-based framework that enables probabilistic upper bounds on the maximum safety violation using only a finite verification set. The following theorem formalizes this finite-sample guarantee:

Theorem 4.4.1 (Safety quantification of Neural CBF)**.** *Consider a continuous time control system (4.1) for which we obtain a Neural CBF $B_\theta(x)$ parameterized by θ. We sample N i.i.d. samples from $\mathcal{X}$. For a user-specified level α, let $\hat{q}$ be the $\frac{\lceil (N+1)(1-\alpha) \rceil}{N}$th quantile of the conformal scores, $s_j = \max_{i \in \{1,2,3\}}(q_i(x_j))$, on the N state samples. Select a safety violation parameter $\epsilon \in (0,1)$ and a confidence parameter $\beta \in (0,1)$ such that:*

$$\mathcal{I}_{1-\epsilon}(N - l + 1, l) \leq \beta, \tag{4.5}$$

where $l = \lfloor (N+1)\alpha \rfloor$ and $\mathcal{I}$ is the regularized incomplete beta function. Then, with the probability of at least $1 - \beta$, the following holds:

$$\mathbb{P}_{x \in \mathcal{X}} \left(\max_{i \in \{1,2,3\}} (q_i(x)) \leq \hat{q} \right) \geq 1 - \epsilon. \tag{4.6}$$

The proof is available in Appendix C.1.1. For a sufficiently high confidence level $(1-\beta)$ and safety level $(1-\epsilon)$, $\hat{q}$ provides a means to quantify the safety of a neural CBF. A positive $\hat{q}$ indicates the presence of safety violations, with its magnitude quantifying the severity of these violations. The procedure to quantify this safety is detailed in Algorithm 2.

Remark 4.4.2. While the above algorithm offers a probabilistic upper bound on the magnitude of safety violations, it does not yield a direct estimate of the value of ψ^*.

In the subsequent subsection, we detail the procedure to estimate ψ^* from the

Algorithm 2 Safety Quantification of Neural CBF

Require: Inputs: Trained NCBF B_θ, induced controller u, dynamics f, g, state domain $\mathcal{X}$, calibration size N, level α

Ensure: Outputs: Conformal score $\hat{q}$, confidence parameter β, violation level ϵ

$D \leftarrow$ Sample N IID states from $\{x : x \in \mathcal{X}\}$

for $i = 0, 1, \ldots, N-1$ **do**

$S_i \leftarrow s_i(0, D)$

end for

$S \leftarrow S$ sorted in decreasing order

$l \leftarrow \lfloor (N+1)\alpha \rfloor$

$\beta, \epsilon \leftarrow N, l$ ▷ From eq. (4.5)

$\hat{q} \leftarrow S_l, \beta, \epsilon$ ▷ From eq. (4.6)

return $\hat{q}, \beta, \epsilon$

conformal score $\hat{q}$ along with the training scheme to synthesize a probabilistically verified NCBF.

4.4.3 Training Procedure

We present the training methodology for synthesizing a Neural Control Barrier Function (NCBF) with probabilistic safety guarantees. At each time step, the controller receives the current system state x and a nominal control input $u_{\text{ref}}(x)$ (any user-supplied nominal policy; it is an input to the QP and is not trained). The goal is to compute a safe control input u that minimally deviates from u_{ref} while satisfying the learned NCBF-based safety constraint. This is achieved by solving a Quadratic Program (NCBF-QP) that filters out unsafe actions:

$$u^\star(x) = \arg\min_{u \in \mathcal{U}} \|u - u_{\text{ref}}(x)\|^2 \quad \text{s.t.} \quad \frac{\partial B_\theta}{\partial x}\big(f(x) + g(x)u\big) + \kappa\big(B_\theta(x)\big) \geq -\psi, \quad (4.7)$$

where the calibrated margin is $\psi = \hat{q}$. Thus, the QP enforces the CBF inequality with right-hand side $-\hat{q}$, consistent with the conformal score definition $q_3(x) \leq \hat{q}$. When $\hat{q} = 0$, constraint (4.7) is the nominal NCBF condition (4.2); a positive score relaxes the inequality, while a negative score tightens it. This remains a marginal, distribution-relative statement rather than a worst-case guarantee over $\mathcal{X}$. The resulting control input $u^\star$ is then utilized in computing the training loss for the subsequent optimization step.

The training dataset is partitioned into mini-batches, and the loss is evaluated batch-wise. Optimization is performed using a stochastic gradient descent method such as ADAM to update the learnable parameter θ. The objective is to learn a parameterization such that the conformal score of the trained NCBF is non-positive, ensuring probabilistic safety.

Algorithm 3 Training Neural CBFs with Probabilistic Safety Assurances

Require: Inputs: Data sets $\mathcal{S}, \mathcal{U}, \mathcal{D}$, dynamics f, g, initialized NCBF parameters θ, calibration size N, level α

Ensure: Output: Trained probabilistically verified NCBF B_θ for deployment through the NCBF-QP

$x_i \leftarrow \text{Sample}(\mathcal{S}, \mathcal{U}, \mathcal{D})$
$\psi \leftarrow 0$
Obtain $\mathcal{L}_\theta(\theta)$. ▷ From eq. (4.4)
while $\mathcal{L}_\theta(\theta) > 0$ **do**
    $B_\theta \leftarrow \theta$
    $\mathcal{L}_\theta(\theta) \leftarrow (B_\theta, f, g, x_i, \psi)$
    $\theta \leftarrow \text{Learn}(\mathcal{L}_\theta, \theta)$
end while
Compute $\hat{q}(N,\, \alpha)$ using Algorithm 2.
$\psi \leftarrow \hat{q}$
Obtain $\mathcal{L}_\theta(\theta)$ using ψ. ▷ From eq. (4.4)
while $\mathcal{L}_\theta(\theta) > 0$ **do**
    $B_\theta \leftarrow \theta$
    $\mathcal{L}_\theta(\theta) \leftarrow (B_\theta, f, g, x_i, \psi)$
    $\theta \leftarrow \text{Learn}(\mathcal{L}_\theta, \theta)$
end while
return B_θ

To this end, we employ an iterative refinement strategy. Initially, the NCBF is trained with $\psi = 0$, as initializing with a nonzero ψ typically induces excessive conservatism. The conformal score $\hat{q}$ of the trained NCBF is then evaluated. If $\hat{q} \leq 0$, the number of violations lies within the user-specified threshold, and ψ can remain zero. Otherwise, ψ is updated as $\psi := \hat{q}$, so that the subsequent training and QP use the calibrated quantile consistently. The NCBF is then retrained with this new ψ. This process is repeated until the conformal score satisfies $\hat{q} \leq 0$ or a predefined maximum number of training iterations is reached. All training is performed offline within a simulated environment. Upon convergence, the resulting NCBF-QP-based controller u can be deployed for online execution on the target system.

This approach provides a scalable and principled way to trade off conservatism with operational flexibility. We detail the procedure to evaluate ψ in Algorithm 3.

4.5 Experiments

We evaluate the effectiveness of our proposed framework on three representative systems: (i) autonomous ground vehicle collision avoidance, (ii) aerial vehicle geofencing, and (iii) quadruped navigation in dynamic environments. Our method is benchmarked

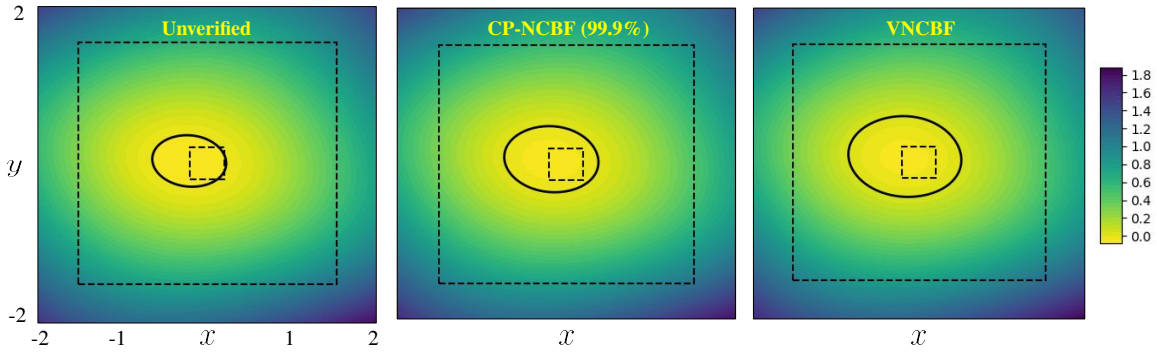


Figure 4.2: Visualization of learned NCBFs for autonomous ground vehicle at $\phi = 0$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (calibrated at a 99.9% marginal coverage level), and (Right) Verified NCBF [165]. Bold black ovals denote the zero-level sets. In this visualization, both **CP-NCBF and Verified NCBF avoid overlapping the unsafe region, while the Unverified NCBF does not**; additionally, CP-NCBF yields **a larger safe region** than the Verified NCBF.

against the verification strategy introduced in [165], which employs Lipschitz regularization and dense sampling to synthesize a verified NCBF. We also compare with an unverified NCBF trained following the approach in [46], in order to investigate the impact of verification. We also analyze the alignment between the probabilistic safety guarantees provided by our method and the observed empirical safety rates during deployment, thereby validating the reliability of the proposed guarantees.

4.5.1 Experimental Case Studies

- **Autonomous Ground Vehicle Collision Avoidance:** In our first experiment, we examine a 3-dimensional collision avoidance problem involving an autonomous ground vehicle governed by Dubins' car dynamics [50]. The objective is to ensure safety by avoiding a static obstacle while navigating through a bounded environment. Further details on the system dynamics, state space bounds, and experimental setup are provided in Appendix C.3.1.

- **Geo-Fencing of Autonomous Aerial Vehicle:** In our next experiment, we investigate a 6-dimensional aerial geo-fencing scenario, wherein an autonomous aerial vehicle is tasked with following a reference trajectory while remaining within a predefined geo-fence region. The safety constraint enforces that the vehicle's state remains confined within this designated region throughout its operation. Further details on the system dynamics, state space bounds, experimental setup and simulation results are provided in Appendix C.3.2.

- **Quadruped navigating in a dynamic environment:** In our final exper-

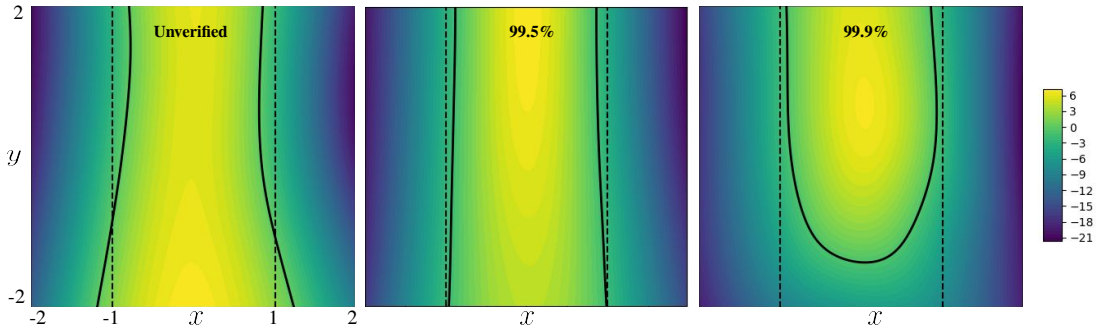


Figure 4.3: Visualization of learned NCBFs for the aerial vehicle at $\phi, \dot{x}_1, \dot{x}_2, \dot{\phi} = 0$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (with **99.5% safety guarantee**), and (Right) CP-NCBF (with **99.9% safety guarantee**). Bold black lines denote the zero-level sets. As the desired **safety guarantees become more stringent**, the learned NCBF **exhibits increased conservatism**. This behavior highlights the flexibility of our proposed framework in enabling a **systematic trade-off between safety and performance, tailored to the specific requirements of the system.**

iment, we evaluate an 8-dimensional quadruped navigation task, wherein a quadruped robot must safely navigate an environment while avoiding a dynamic obstacle. This scenario introduces significant complexity due to the time-varying nature of the obstacle, which poses challenges in enforcing safety constraints and increases the dimensionality of the underlying dynamical system. This setting also serves as a representative benchmark to assess the scalability of our proposed framework to high-dimensional, safety-critical control tasks. Further details on the system dynamics, state space bounds, experimental setup and simulation results are provided in Appendix C.3.3.

4.5.2 Comparative Analysis

Figure 4.2 presents a comparative analysis of the zero-level sets for the autonomous ground vehicle collision avoidance task at $\phi = 0$, across three approaches: (left) an unverified NCBF trained without robustness considerations, (center) the proposed Conformal NCBF (CP-NCBF) offering a 99.9% probabilistic safety guarantee, and (right) a verified NCBF synthesized using the method of [165]. The unverified NCBF exhibits overlap with the predefined unsafe set, revealing safety violations likely induced by insufficient generalization. In contrast, both the CP-NCBF and the verified NCBF avoid unsafe regions, highlighting the necessity of incorporating verification during NCBF synthesis.

A similar trend is observed in the aerial vehicle geo-fencing task depicted in Figure 4.3, further validating the efficacy of safety-aware training. Importantly, the CP-

NCBF yields a considerably less conservative approximation of the safe set compared to [165], while still maintaining high safety levels (99.9%), thereby demonstrating an improved trade-off between conservatism and performance. Figure 4.3 also illustrates that tightening the safety guarantees leads to increasingly conservative safe regions. This characteristic enables practitioners to systematically adjust the degree of conservatism to suit application-specific safety-performance trade-offs. Thus, our framework facilitates flexible and principled tuning of control policies under safety constraints.

These findings generalize to higher-dimensional tasks as well. Figure C.4 (Appendix C.3.3) illustrates similar trends in the quadruped navigation setting, validating the scalability and practical relevance of our framework for deployment in complex, high-dimensional, safety-critical systems.

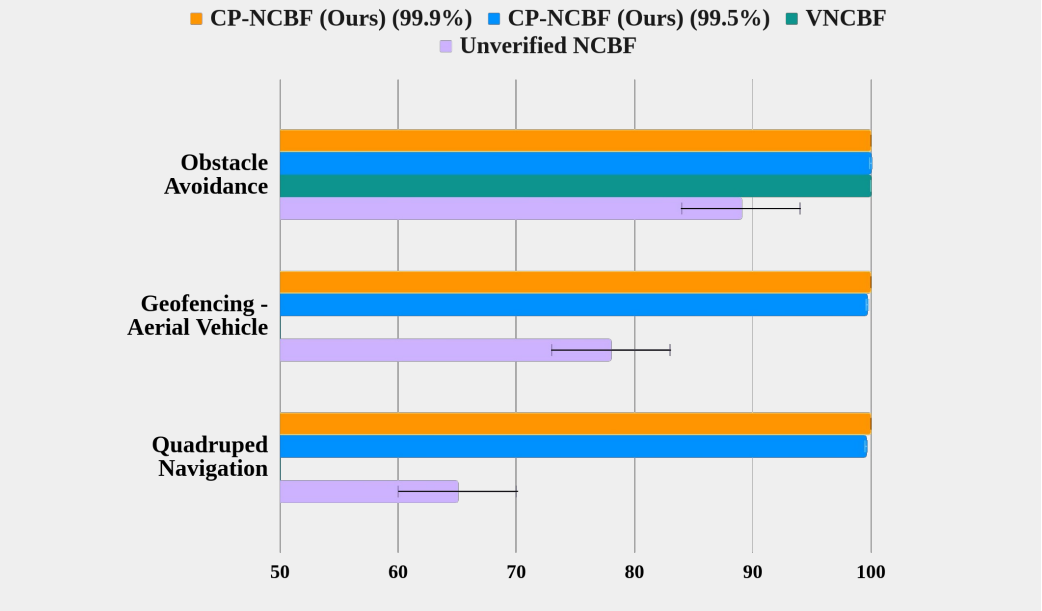


Figure 4.4: This figure presents a comparative analysis of the empirical safety rates achieved by all evaluated methods. **The observed safety rates align closely with the corresponding theoretical guarantees**, thereby providing empirical validation of the reliability and robustness of the proposed framework. These findings highlight the potential of our approach for **deployment in real-world safety-critical applications**. We also note that **VNCBF could not be computed** for the geo-fencing and quadruped navigation tasks due to its **reliance on grid-based sampling, which is computationally feasible only for systems up to** 5 **dimensions**. This limitation further illustrates the **scalability advantage of our method, which remains tractable and effective even in high-dimensional settings.**

Lastly, Figure 4.4 summarizes the empirical safety rates attained by all evaluated methods, aggregated over five random seeds with error bars representing the corresponding variance. Notably, the VNCBF baseline could not be computed for the geofencing and quadruped navigation tasks due to computational intractability,

thereby reinforcing our claim regarding the scalability advantages of the proposed verification framework in high-dimensional settings. The observed empirical safety rates are in close agreement with the theoretical guarantees, highlighting both the reliability and practical applicability of our probabilistic verification framework in safety-critical autonomous systems.

4.6 Chapter Summary

We propose CP-NCBF, a framework for synthesizing neural CBFs with probabilistic safety guarantees via split conformal prediction. Our method bridges the gap between learning-based synthesis and formal safety verification, enabling the construction of less conservative safe sets with improved sample efficiency. Case studies on autonomous driving and aerial vehicle geo-fencing tasks validate that our method achieves real-time safety and broadens the safe operational domain compared to existing approaches.

Limitations and Future Work: The current work assumes the knowledge of the dynamics (f and g). The methodology currently only focuses on safety, without any focus on performance objectives. As a part of future work, we plan to extend the framework to account for unknown dynamics and also work on synthesizing NCBFs to co-optimize both safety and performance. It would also be exciting to extend this framework to more complex, high-dimensional systems.

Two further aspects of the guarantees deserve explicit comment. First, the conformal certificate is *probabilistic by design*: it states that at most a fraction ϵ of states violate the CBF conditions (with confidence $1 - \beta$), and hence it does not rule out violations entirely, in contrast to the set-invariance guarantees of Chapter 3. In exchange, it remains valid with finite samples and without global Lipschitz bounds. Practitioners can manage this residual risk by selecting ϵ and β conservatively for the application at hand (e.g., 10^{-4} and 10^{-10} in our studies), monitoring violations online, and falling back to robust-CBF-style tightening [191] when the deployment environment is less benign than the calibration distribution. Second, all training and calibration are performed offline on a nominal model. The sim-to-real gap is therefore not captured by the certificate: if the physical system deviates from the model used to generate the scores, the effective violation rate may exceed ϵ. Mitigations include calibrating $\hat{q}$ on hardware-collected state data whenever available, domain randomization during sampling so that the calibration distribution covers realistic variability, and the graceful degradation inherent to the QP filter (4.7), which remains feasible and merely loses its tightness under moderate mismatch. A formal treatment of this

gap—e.g., conformal calibration over a distribution of model perturbations—is an exciting direction for future work.

Regarding training assumptions, we note that the offline, time-invariant formulation does not preclude *moving* obstacles: in the quadruped study, the obstacle position, velocity, and radius are folded into the augmented state $\mathcal{X} \subseteq \mathbb{R}^8$, safe/unsafe labels are generated via collision cones over that augmented space, and the QP (4.7) is re-evaluated online at each measured obstacle state. Safety is thereby enforced against the obstacle trajectory realizations represented in the calibration distribution; genuinely adversarial or unmodeled obstacle motion falls under the sim-to-real caveat above.

Finally, a comparison of data requirements across the three synthesis paradigms developed thus far is instructive. The Lipschitz-verified approach of Chapter 3 requires dense sampling of $\mathcal{X}$ together with LMI-enforced global bounds, and its grid-based verification scales exponentially with dimension (it could not be computed beyond $n = 5$ in our benchmarks). CP-NCBF requires two finite datasets—N_{train} training samples and N i.i.d. verification points for calibration—both independent of the grid resolution; empirically, 20K training points suffice for $n = 3$–6 while the quadruped task uses 50K for $n = 8$ (Appendix C.4), and Appendix C shows the conformal scores decreasing steadily with training-set size, quantifying the sample-efficiency trade-off. The unverified baseline requires only training samples but offers no guarantee; CP-NCBF thus sits between the two, adding one calibrated dataset in exchange for distribution-free probabilistic assurance and markedly better scalability than Lipschitz-based verification.

Bridge to the Next Chapter. Chapters 3 and 4 have developed learning-based safety certificates (Lipschitz-verified and conformal, respectively) for deterministic control-affine systems. Both verification paradigms provide formal guarantees from finitely many samples but are limited to the deterministic setting. In practice, many safety-critical systems are subject to stochastic disturbances (e.g., sensor noise, wind gusts, unmodelled dynamics), requiring barrier certificates that account for diffusion terms.

The next chapter addresses this gap by extending the NCBF framework to stochastic systems governed by SDEs. Chapter 5 develops Verification-In-The-Loop (VITL) synthesis of Stochastic Neural CBFs, combining NLP-based counterexample verification with iterative training to produce probabilistically safe barrier certificates for both smooth and ReLU-activated networks.

Chapter 5

Verification-in-the-Loop Synthesis of Stochastic Neural CBFs

5.1 Related Works

Safety, as a fundamental requirement in control systems, is typically formulated as the positive invariance of a given set of states [19], called a safe set. It ensures that the state trajectories remain within a safe set for all time. Several methodologies have been proposed to guarantee safety. Hamilton-Jacobi reachability analysis [12, 160, 33] is a model-based method to determine the maximal safe set of states for a dynamical system. By solving a Hamilton-Jacobi-Bellman (HJB) partial differential equation (PDE) or Hamilton-Jacobi-Isaacs (HJI) variational inequality (for systems with disturbances), one obtains a value function whose zero sublevel set represents all states from which the system can be driven to avoid entering an unsafe region. However, a well-known drawback of HJ reachability is the curse of dimensionality: solving the HJ PDE requires gridding the entire state space, which increases computational demand exponentially as the state dimension increases [13].

Another line of research focuses on learning-enabled safe control, including data-driven control and reinforcement learning (RL). These approaches integrate safety constraints into RL or NN-based controllers and address safety through constrained policy optimization [24] or risk-sensitive objectives [36]. Recent hybrid architectures combine RL with formal verification tools [31] or backup controllers [184], yet provable safety guarantees under distribution shifts remain limited. Sample efficiency and sim-to-real gaps in learning-enabled safe control further hinder deployment in safety-critical hardware applications.

Control barrier functions (CBFs) have emerged as a computationally efficient so-

lution for safety-critical control, enforcing set invariance through quadratic program (QP) formulations [7]. CBF-QP is an efficient and reliable approach to serve as a safety filter [6]. For systems subject to random disturbances, Stochastic CBFs are proposed to ensure probabilistic guarantees and synthesize safe control signals [73, 38]. In many applications, CBFs are not always given and are difficult to derive. For polynomial CBFs, Sum-of-squares (SOS) optimization is a popular tool for CBF synthesis. Derived from algebraic geometry, SOS optimization has been widely used for synthesizing barrier certificates [121], safety index functions [203], and CBFs [41, 42, 136].

The success of SOS optimization for CBFs depends on the ability to represent safety constraints as polynomials, however, complex safety constraints cannot always be encoded as polynomials. Neural barrier certificates [1, 124, 125] and Neural Control Barrier Functions (NCBFs) [46, 44, 98] have been proposed to harness the expressiveness of neural networks. The synthesis of NCBFs faces key challenges during the training of their representative NNs [145]. Existing work primarily focuses on deterministic systems and leaves stochastic NCBFs less studied. SNCBFs were first investigated in [195], where a fault-tolerant SNCBF was created by stacking SNCBFs under different sensor fault patterns. However, the training of SNCBFs requires additional verification to check if the synthesized SNCBF is both correct and feasible for the given system dynamics and safety constraints. In this work, we propose SNCBF synthesis frameworks to synthesize verifiably safe SNCBFs.

NCBF verification combines two challenging problems, namely, the input-output verification of neural networks (VNN) [200, 189, 134, 190, 180, 199] and the reachability verification of nonlinear systems. The requirement for NCBF verification makes VNN approaches not directly applicable. Incorporating VNN into NCBF verification [198] or formulating NCBF verification into a VNN problem [71] are non-trivial tasks. While sound and complete verifiers such as dReal can be applied to NCBFs, they typically can only handle low-dimensional systems or small neural networks [1, 51]. In [198], exact conditions for safety verification of NCBFs with ReLU activation functions were proposed that leverage the piecewise-linearity of ReLU-NNs for better efficiency. Similarly, [196] decomposed the ReLU NCBF and proposed a hierarchical efficient verification algorithm for efficient SMT verification. However, these approaches remain on NCBFs for deterministic systems, leaving NCBFs for stochastic systems less studied. This chapter proposes the first SNCBF verification that can be solved by nonlinear programming or SMT solvers.

Our prior conference work [165] focuses on smooth SNCBF synthesis but leaves ReLU, one of the most popular NN activation functions, unexplored. In this chapter, we introduce ReLU SNCBFs and propose synthesis algorithms for them. We derive

ReLU NN verification conditions and propose a practical and efficient algorithm for verification.

5.2 Contributions of this Chapter

This chapter makes the following contributions towards synthesizing and verifying Neural CBFs for stochastic systems:

1. We formulate the verification of smooth SNCBFs with twice-differentiable activation functions as nonlinear programs that can be solved by Satisfiability Modulo Theories (SMT) solvers.
2. We introduce the Stochastic NCBF (SNCBF) with ReLU activation functions and derive sufficient safety conditions using Tanaka's formula.
3. We utilize the piecewise linearity of ReLU NNs, formulate the verification of ReLU SNCBFs as nonlinear programs, and propose practical algorithms for efficient verification.
4. We frame the Verification-In-The-Loop (VITL) synthesis with efficient verifiers for both smooth and ReLU NCBF synthesis. The VITL relaxes the dense sampling and single-layer assumption of the smooth SNCBF.
5. We validate our approach on three case studies, namely the inverted pendulum, Darboux, and the unicycle model, demonstrating that both smooth and ReLU SNCBFs produce verifiably safe results while covering a larger safe subset compared to the baseline Fault-Tolerant SNCBF without VITL.

The results presented in this chapter are based on [165, 197].

5.3 Preliminaries

In this section, we introduce the system model, stochastic control barrier functions (SCBFs), and needed mathematical background.

5.3.1 System Model

We consider a continuous time stochastic control system with state $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and input $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ at time $t \geq 0$ with a dynamic model formulated by the following stochastic differential equation (SDE):

$$d\mathbf{x}_t = (f(\mathbf{x}_t) + g(\mathbf{x}_t)u_t)\ dt + \sigma\ dW_t, \tag{5.1}$$

where $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$ are locally Lipschitz, W_t is n_v-dimensional Brownian motion and $\sigma \in \mathbb{R}^{n_x \times n_v}$ is the diffusion matrix. Throughout this chapter, the symbol σ (i.e., σ) is reserved exclusively for this diffusion matrix; neural-network activation functions are denoted by φ, to keep the two notions notationally distinct. We make the following assumptions on system (5.1).

Assumption 5.1. *There exists an initial state $\hat{\mathbf{x}} \in \mathcal{X}$ such that $\mathbb{P}[\mathbf{x}_0 = \hat{\mathbf{x}}] = 1$. The SDE (5.1) admits a unique strong solution.*

We denote the state space without obstacles as a safe set $\mathcal{X}_S$ and let $\mathcal{X}_U = \mathcal{X}/\mathcal{X}_S$ be the unsafe set. The safe set is defined as the *super-0-level set* of a differentiable function $h : \mathcal{X} \subseteq \mathbb{R}^{n_x} \to \mathbb{R}$, yielding

$$\mathcal{X}_S = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) \geq 0\} \tag{5.2}$$

$$\mathcal{X}_U = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) < 0\}. \tag{5.3}$$

We further let the interior and boundary of $\mathcal{X}_S$ be $\text{Int}\,(\mathcal{X}_S) = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) > 0\}$ and $\partial\mathcal{X}_S = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) = 0\}$, respectively.

Function Definitions: We denote the *indicator function* as $\mathbb{1}_{\mathcal{C}}(\mathbf{x})$ which takes the value of 1 if $\mathbf{x} \in \mathcal{C}$ and 0 if $\mathbf{x} \notin \mathcal{C}$. Let ω denote a scalar variable and a denote a constant scalar. We use the indicator function for logical operations, i.e., $\mathbb{1}_{\omega > a}(x)$, which takes the value of 1 when $\omega > a$. Let $\{\omega - a\}_+ := \max\{\omega - a, 0\}$ and $\{\omega - a\}_- := -\min\{\omega - a, 0\}$. For comparison function definitions (class-κ, extended class-κ_∞), we refer the reader to Chapter 2.

Notations: We denote the stochastic process of the state $\mathbf{x}$ as $\mathbf{x}_t$ and let $\mathbf{x}_0$ represent the initial state. Let $[\cdot]_i$ denote the i-th row of a matrix or the i-th item of a vector. For simplicity, we use $= \mathbf{0}$, and $\geq \mathbf{0}$ to represent the vector element-wise $= 0$, and ≥ 0, respectively.

5.3.2 Preliminaries on Stochastic Processes

We next present concepts from stochastic processes, namely, Tanaka's formula and the infinitesimal generator, which we will use later to derive the safety conditions for our ReLU SNCBF.

Any strong solution of an SDE is a semimartingale. Let ω_t denote a scalar continuous semimartingale process. The following Tanaka's formula explicitly links the local time of a semimartingale to its behavior.

Theorem 5.3.1 (Tanaka's Formula [131], Ch. 6, Theorem 1.2)**.** *For any real number a, there exists an increasing continuous process L_t^a called the local time of ω_t in a, such that,*

$$|\omega_t - a| = |\omega_0 - a| + \int_0^t \operatorname{sgn}(\omega_s - a)\, d\omega_s + L_t^a$$
$$\{\omega_t - a\}_+ = \{\omega_0 - a\}_+ + \int_0^t \mathbb{1}_{(\omega_s > a)} d\omega + \frac{1}{2} L_t^a$$
$$\{\omega_t - a\}_- = \{\omega_0 - a\}_- - \int_0^t \mathbb{1}_{(\omega_s \leq a)} d\omega_s + \frac{1}{2} L_t^a.$$

A detailed treatment of semimartingales and Tanaka's formula can be found in [131]. Consider a function $B : \mathbb{R}^{n_x} \to \mathbb{R}$ mapping from a state $\mathbf{x}$ to a scalar. The infinitesimal generator of B is defined as follows.

Definition 5.3.2 (Infinitesimal generator)**.** Let $\mathbf{x}_t$ be the strong solution to (5.1). The infinitesimal generator $\mathcal{A}$ of $\mathbf{x}_t$ is defined by

$$\mathcal{A}B(\mathbf{x}) = \lim_{t \downarrow 0} \frac{\mathbb{E}\left[B\left(\mathbf{x}_t\right) \mid \mathbf{x}_0 = \mathbf{x}\right] - B(\mathbf{x})}{t}; \quad \mathbf{x} \in \mathbb{R}^{n_x}$$

where $B(\mathbf{x})$ is in a set of functions $\mathcal{D}(\mathcal{A})$ (called the domain of the operator $\mathcal{A}$) such that the limit exists at $\mathbf{x}$.

We make the following assumption on $B(\mathbf{x})$.

Assumption 5.2. *We assume the function $B(\mathbf{x})$ is in the domain of the operator $\mathcal{D}(\mathcal{A})$. Specifically, the following two conditions are satisfied in $\mathcal{D}(\mathcal{A})$.*

- *The expectation $\mathbb{E}\left[B\left(\mathbf{x}_t\right) \mid \mathbf{x}_0 = \mathbf{x}\right]$ is well-defined and finite.*
- *The limit defining $\mathcal{A}B(\mathbf{x})$ exists for every $\mathbf{x} \in \mathcal{X}$.*

We note that $\mathcal{D}(\mathcal{A}) = \mathbb{R}^{n_x}$ if $B(\mathbf{x})$ is twice differentiable.

5.3.3 Stochastic Control Barrier Functions

The control policy must ensure that the system adheres to a safety constraint defined by the positive invariance of a specified safety region $\mathcal{X}_S$.

Control Barrier Functions for Stochastic Systems

We introduce the stochastic CBF (SCBF) for stochastic systems by presenting the definition of stochastic (zeroing) CBF and SCBF-QP.

Definition 5.3.3 (Stochastic Zeroing CBF)**.** A continuously differentiable function $B : \mathcal{X} \to \mathbb{R}$ is called a stochastic zeroing control barrier function for system (5.1) if the following conditions are satisfied.

- $B(\mathbf{x}) \in \mathcal{D}(\mathcal{A})$;
- $B(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathcal{D} \subseteq \mathcal{X}_S$;
- $B(\mathbf{x}) < 0$ for all $\mathbf{x} \notin \mathcal{D}$;
- there exists an extended κ_∞ function α such that

$$\sup_{u \in \mathcal{U}} [\mathcal{A}B(\mathbf{x}) + \alpha(B(\mathbf{x}))] \geq 0.$$

Let $\mathcal{D} := \{\mathbf{x} : B(\mathbf{x}) \geq 0\}$ to denote the super-0-level-set of the function $B(\mathbf{x})$.

The SCBF-QPs are regarded as *safety filters* which take the control input from a reference controller $u_{ref}(\mathbf{x}, t)$ and modify it so that it abides by our constraint of positive invariance:

$$\begin{aligned} u^*(\mathbf{x}, t) = \min_{u \in \mathcal{U}} & \|u - u_{ref}(\mathbf{x}, t)\|^2 \\ \text{s.t. } & \mathcal{A}B(\mathbf{x}) + \alpha(B(\mathbf{x})) \geq 0. \end{aligned} \tag{5.4}$$

Control signals satisfying SCBF-QP ensure probabilistic positive invariance of the set $\mathcal{D}$.

Next, we present the worst-case probability estimation.

Proposition 5.3.4 ([177, Proposition III.5])**.** *Suppose the map $B(\mathbf{x})$ is a Stochastic CBF with a linear class-α function (where $\alpha(\mathbf{x}) = k\mathbf{x}; k > 0$), and the control strategy as $\mathcal{U}_z = \{u \in \mathcal{U} : \mathcal{A}B(\mathbf{x}) + kB(\mathbf{x}) \geq 0\}$. Let $c = \sup_{x \in \mathcal{C}} B(\mathbf{x})$ and $\mathbf{x}_0 \in int(\mathcal{D})$, then under any $u \in \mathcal{U}_z$ we have the following worst-case proba-*

bility estimation:

$$\mathbb{P}\left[\mathbf{x}_t \in int(\mathcal{D}), 0 \leq t \leq T\right] \geq \left(\frac{B(\mathbf{x}_0)}{c}\right) e^{-cT}.$$

Neural CBFs

CBFs that are defined by neural networks, denoted as Neural Control Barrier Functions (NCBFs), have been proposed to harness the expressiveness of NNs. We consider a θ-parameterized feedforward neural network $B_\theta : \mathbb{R}^{n_x} \to \mathbb{R}$ constructed as follows. The network consists of L layers, with each layer i consisting of M_i neurons. We let $(i, j) \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}$ denote the j-th neuron at the i-th layer. We let $\mathbf{W}$ and $\mathbf{r}$ denote the weight and bias of a neural network, and let θ be a parameter vector obtained by concatenating $\mathbf{W}$ and $\mathbf{r}$. We denote the pre-activation input of node j in layer i as $z_j^{(i)}$ and the activation function φ.

An NCBF synthesized for a stochastic system is referred to as SNCBF. In this chapter, we consider two types of SNCBFs: smooth SNCBFs and ReLU SNCBFs. An SNCBF is a smooth SNCBF if all its activation functions $\varphi(\cdot)$ are smooth functions, i.e., $\tanh(\mathbf{x})$, *Sigmoid*, *Softmax*. An SNCBF is a ReLU SNCBF if all its activation functions are ReLU, i.e., $\varphi(\mathbf{x}) = \max\{\mathbf{x}, 0\}$. For simplicity, we use the short-hand notation $B(\mathbf{x})$ to denote B_θ for the rest of this chapter.

The piece-wise linearity of ReLU NN allows us to have a linear expression to represent the input-output relationship of the NN with the activated neurons. We denote $\mathbf{z}_i$ as the pre-activation input vector of the i-th layer.

$$z_{ij}(\mathbf{x}) = \begin{cases} W_{1j}^T \mathbf{x} + r_{1j}, & i = 1 \\ W_{ij}^T \varphi(\mathbf{z}_{i-1}(\mathbf{x})) + r_{ij}, & i \in \{2, \ldots, L-1\} \end{cases}$$

where φ is an elementary activation function. We denote $z_{ijt} = z_{ij}(\mathbf{x}_t)$. The output of the network is given by $B(\mathbf{x}) = W_L^T \mathbf{z}_L + r_L$. The j-th neuron at the i-th layer is *activated* by a particular input $\mathbf{x}$ if its pre-activation input is positive, *inactivated* if the pre-activation input is negative, and *unstable* if the pre-activation input is zero. An *activated set* $\mathbf{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_L\}$ denotes the set of neurons $\mathcal{S}_i \subseteq \{1, \ldots, M_i\}$ that are activated at the i-th layer. For the first layer, we have

$$\overline{W}_{1j}(\mathbf{S}) = \begin{cases} W_{1j}, & j \in S_1 \\ 0, & \text{else} \end{cases} \quad \overline{r}_{1j}(\mathbf{S}) = \begin{cases} r_{1j}, & j \in S_1 \\ 0, & \text{else.} \end{cases}$$

We recursively define $\overline{W}_{ij}(\mathbf{S})$ and $\overline{r}_{ij}(\mathbf{S})$ by letting $\overline{\mathbf{W}}_i(\mathbf{S})$ be a matrix with columns

$\overline{W}_{i1}(\mathbf{S}), \ldots, \overline{W}_{iM_i}(\mathbf{S})$.

The ReLU NN can be decomposed into hyperplanes and hinges. The hyperplane characterized by an activated set $\mathbf{S}$ is defined as [198, Lemma 1]

$$\mathcal{X}(\mathbf{S}) \triangleq \bigcap_{i=1}^{L} \left(\bigcap_{j \in \mathcal{S}_i} \{\mathbf{x} : W_{ij}^T(\overline{\mathbf{W}}_{i-1}(\mathbf{S})\mathbf{x} + \overline{\mathbf{r}}_{i-1}) + r_{1j} \geq 0\} \right.$$
$$\left. \cap \bigcap_{j \notin \mathcal{S}_i} \{\mathbf{x} : W_{ij}^T(\overline{\mathbf{W}}_{i-1}(\mathbf{S})\mathbf{x} + \overline{\mathbf{r}}_{i-1}) + r_{1j} \leq 0\} \right). \quad (5.5)$$

We denote the hinges, the regions at the intersections of hyperplanes with unstable neurons, by $\mathbf{T}$. For example, the hinge characterized by $\mathbf{S}_1, \ldots, \mathbf{S}_r$ is defined as follows.

$$\mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) := \left(\bigcup_{l=1}^{r} \mathcal{X}(\mathbf{S}_l) \right) \setminus \left(\bigcap_{l=1}^{r} \mathcal{X}(\mathbf{S}_l) \right).$$

We write that $\mathbf{S}_1, \ldots, \mathbf{S}_r$ is *complete* if for any $\mathbf{T}' \subseteq \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$, $(\bigcap_{l=1}^{r} \mathcal{X}(\mathbf{S}_l)) \cup \mathbf{T}' \subseteq \{\mathcal{X}(\mathbf{S}_1), \ldots, \mathcal{X}(\mathbf{S}_r)\}$.

> *Remark* 5.3.5 (Verification versus Certification). The terms *verification* and *certification* are used deliberately and distinctly throughout this chapter. *Verification* refers to the algorithmic procedure—solving the nonlinear programs derived from the Positivstellensatz and Farkas' Lemma, or enumerating ReLU activation regions—that checks whether the SNCBF conditions hold over every region of the network. *Certification* refers to the resulting mathematical guarantee attached to the trained pair (B, u) when verification succeeds. The two are linked but not identical: a successful verification run yields a certificate, a failed run yields a counterexample that feeds back into training, and any certification claim inherits exactly the assumptions under which the verifier was constructed (e.g., the Lipschitz bounds of Assumption 5.3 or completeness of region enumeration).

5.3.4 Preliminary Results

We next present preliminary results from Positivestellensatz and Farkas' Lemma. These results will be used to derive the verification conditions for SNCBFs. The polynomial $p(\mathbf{x})$ is Sum-Of-Squares (SOS) if and only if it can be written as

$$p(\mathbf{x}) = \sum_{i=1}^{n} (p_i(\mathbf{x}))^2$$

for some polynomials $p_i(\mathbf{x})$.

We next present the required background on real algebraic geometry. For certifying non-negative polynomials, the cone is generated from a set of polynomials $\phi_1, \ldots, \phi_{k_\Sigma}$ given by

$$\Sigma[\phi_1, \ldots, \phi_{k_\Sigma}] = \left\{ \sum_{K \subseteq \{1,\ldots,k_\Sigma\}} \gamma_K(\mathbf{x}) \prod_{i \in K} \phi_i(\mathbf{x}) : \gamma_K(\mathbf{x}) \in \text{ SOS } \forall K \subseteq \{1, \ldots, k_\Sigma\} \right\}.$$

The monoid $\mathcal{M}$ generated from a set of polynomials $\chi_1, \ldots, \chi_{k_\mathcal{M}}$ with non-negative integer exponents $r_1 \ldots r_{k_\mathcal{M}}$ is defined as

$$\mathcal{M}[\chi_1, \ldots, \chi_{k_\mathcal{M}}] = \left\{ \prod_{i=1}^{k_\mathcal{M}} \chi^{r_i}(\mathbf{x}) : r_1 \ldots r_{k_\mathcal{M}} \in \mathbb{Z}_+ \right\}.$$

The ideal generated from polynomials $\Gamma_1, \ldots, \Gamma_{k_\mathbb{I}}$ is given by

$$\mathbb{I}[\Gamma_1, \ldots, \Gamma_{k_\mathbb{I}}] \quad = \quad \left\{ \sum_{i=1}^{k_\mathbb{I}} p_i(\mathbf{x}) \Gamma_i(\mathbf{x}) : p_1, \ldots, p_{k_\mathbb{I}} \text{ are polynomials} \right\}.$$

The following theorem shows that the emptiness of a set formed by the polynomials described above is equivalent to the existence of polynomials within their respective cone, square of monoid, and ideal such that their sum is equal to zero.

Theorem 5.3.6 (Positivstellensatz [116]). *Let $(\phi_i)_{j=1,\ldots,k_\Sigma}$, $(\chi_j)_{k=1,\ldots,k_\mathcal{M}}$, $(\Gamma_\ell)_{\ell=1,\ldots,k_\mathbb{I}}$ be finite families of polynomials. Then, the following properties are equivalent:*

1. *The set*

$$\left\{ \mathbf{x} \in \mathbb{R}^{n_x} \;\middle|\; \begin{array}{ll} \phi_i(\mathbf{x}) \geq 0, & i = 1, \ldots, k_\Sigma \\ \chi_j(\mathbf{x}) \neq 0, & j = 1, \ldots, k_\mathcal{M} \\ \Gamma_l(\mathbf{x}) = 0, & l = 1, \ldots, k_\mathbb{I} \end{array} \right\} \tag{5.6}$$

   is empty.

2. *There exist $\phi \in \Sigma, \chi \in \mathcal{M}, \Gamma \in \mathbb{I}$ such that $\phi + \chi^2 + \Gamma = 0$.*

The following is one of the variants of Farkas' Lemma.

Lemma 5.1 (Farkas' Lemma [106, Proposition 6.4.3)]). *Let A be a real matrix with m rows and n columns, and let $b \in \mathbb{R}^n$ be a vector. Exactly one of the following conditions holds (the conditions are mutually exclusive):*

- *The system of inequalities $Ax \leq b$ has a solution*

- *There exists y such that $y \geq \mathbf{0}$, $y^T A = \mathbf{0}^T$ and $y^T b < 0$, where $\mathbf{0}$ denote a zero vector.*

5.4 Smooth Stochastic Neural Control Barrier Functions

In this section, we present the synthesis of SNCBF with smooth activation functions. Our approach assumes that the NN employs smooth (i.e., twice differentiable) activation functions so that its Jacobian and Hessian are well-defined. The synthesis procedure is developed in two frameworks: (i) verifiable synthesis and (ii) VITL synthesis with a novel verification for smooth SNCBFs.

Problem 5.1. *Given a continuous-time stochastic control system defined as (5.1), a desired confidence level η, state space $\mathcal{X}$, initial safe set and the unsafe set $\mathcal{X}_I \subseteq \mathcal{X}_S$ and $\mathcal{X}_U = \mathcal{X} \backslash \mathcal{X}_S$, respectively, devise an algorithm that synthesizes a stochastic neural CBF* (SNCBF) *$B(\mathbf{x})$ with continuously differentiable (smooth) activation functions. In particular, the SNCBF must satisfy the following worst-case probability guarantee:*

$$\mathbb{P}\left[\mathbf{x}_t \in \mathcal{X}_S, \forall t \in [0, T] \mid \mathbf{x}_0 \in int(\mathcal{D})\right] \geq 1 - \eta.$$

Our strategy to address Problem 5.1 is to (i) perform a verifiable synthesis of the SNCBF under a set of initial assumptions, (ii) relax some of these assumptions by introducing verification conditions, and (iii) propose a synthesis loss function that embeds verification into the learning loop.

To facilitate the synthesis and verification of an SNCBF that meets the safety criteria in Problem 5.1, it is essential to analyze how the barrier function evolves along the trajectories of the stochastic system. In continuous-time settings, the following lemma provides the precise expression for the infinitesimal generator of an SNCBF with smooth activation functions:

The infinitesimal generator of an SNCBF (SNCBF) $B(\mathbf{x})$ with twice-differentiable activation functions is given by [141] as:

$$\mathcal{A}B = \frac{\partial B}{\partial \mathbf{x}}\left(f(\mathbf{x}) + g(\mathbf{x})u\right) + \frac{1}{2}\mathsf{tr}\left(\sigma^{\intercal}\frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma\right). \tag{5.7}$$

5.4.1 Smooth SNCBF Verifiable Synthesis

The conditions of Problem 5.1 are satisfied if we can synthesize a NN $B(\mathbf{x})$ parameterized by θ to represent our SCBF. We denote control policy $\mu : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ as a mapping from the state $\mathbf{x}_t$ to a control input $u_t = \mu(\mathbf{x}_t)$ at each time t. $B(\mathbf{x})$ ensures safety by imposing conditions on system states and control inputs defined as follows.

$$
\begin{aligned}
&B(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathcal{X}_I, \\
&B(\mathbf{x}) < 0, \forall \mathbf{x} \in \mathcal{X}_U, \\
&\frac{\partial B(\mathbf{x})}{\partial \mathbf{x}}(f(\mathbf{x}) + g(\mathbf{x})u) + \frac{1}{2}\mathsf{tr}\left(\sigma^\intercal \frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma\right) + \alpha\left(B(\mathbf{x})\right) \geq 0, \forall \mathbf{x} \in \mathcal{D}.
\end{aligned} \tag{5.8}
$$

To approach Problem 5.1, we make the following assumptions in Section 5.4.1.

Assumption 5.3. *For the smooth SNCBF verifiable synthesis, we assume that*

1. *The function $B(\mathbf{x})$, $\frac{\partial B}{\partial \mathbf{x}}$, $\frac{\partial^2 B}{\partial \mathbf{x}^2}$ and control policy μ are locally Lipschitz continuous.*
2. *The SNCBF $B(\mathbf{x})$ is a feedforward NN with one hidden layer and twice differentiable activation function.*
3. *σ is a $n_x \times n_x$ diagonal matrix.*
4. *The derivative of each activation function is lower bounded by $\underline{\varphi'}$ and upper bounded by $\overline{\varphi'}$.*

Remark 5.4.1 (Interpretation of the Smoothness Assumptions). Assumption 5.3(1) and (4) may appear stronger than the deterministic counterpart in Chapter 3, which required Lipschitz continuity only of B and $\frac{\partial B}{\partial \mathbf{x}}$. The additional requirement stems directly from the Itô correction term $\frac{1}{2}\mathsf{tr}\left(\sigma^\intercal \frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma\right)$ in (5.7): since q_3 in (5.10) contains the Hessian of B, propagating sample-level satisfaction across a cover of $\mathcal{X}$ requires bounding how fast that curvature-driven term can vary, i.e., a Lipschitz bound on $\frac{\partial^2 B}{\partial \mathbf{x}^2}$. For the single-hidden-layer networks considered here, this bound is enforced constructively via the LMI certificate on $M_{\bar{\varphi}}(\bar{\theta}, \Omega)$ of Section 5.4.1, so the assumption holds by training-time construction rather than as an external restriction. Assumption 5.3(3) (diagonal σ) is made for notational convenience and entails no loss of generality whenever the noise channels are independent.

Next, we formulate our verifiable synthesis of $B(\mathbf{x})$ as a robust optimization problem (ROP) with auxiliary variable ψ:

$$\text{ROP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & \max\left(q_k(\mathbf{x})\right) \leq \psi, k \in \{1,2,3\} \\ & \forall \mathbf{x} \in \mathcal{X}, \psi \in \mathbb{R}, \end{cases} \tag{5.9}$$

where

$$\begin{aligned} q_1(\mathbf{x}) &= \left(-B(\mathbf{x})\right) \mathbb{1}_{\mathcal{X}_I}(\mathbf{x}), \\ q_2(\mathbf{x}) &= \left(B(\mathbf{x}) + \delta\right) \mathbb{1}_{\mathcal{X}_U}(\mathbf{x}), \\ q_3(\mathbf{x}) &= -\frac{\partial B}{\partial \mathbf{x}}\left(f(\mathbf{x}) + g(\mathbf{x})u\right) - \frac{1}{2}\mathsf{tr}\left(\sigma^{\intercal}\frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma\right) - \alpha(B(\mathbf{x})), \end{aligned} \tag{5.10}$$

where δ is a small positive scalar ensuring strict inequality. The ROP in (5.9) inherently involves infinitely many constraints due to the continuous nature of the state space $\mathcal{X}$.

To approximate the solution, we employ the scenario optimization program (SOP), which replaces the ROP with a finite number of sampled constraints. Given a scalar $\bar{\epsilon}$, we uniformly sample N data points $\mathbf{x}_i \in \mathcal{X}, i \in \{1, \ldots, N\}$. Let $\bar{\epsilon}$ be a positive scalar. We sample points dense enough such that $\|\mathbf{x} - \mathbf{x}_i\| \leq \bar{\epsilon}$ for any $\mathbf{x} \in \mathcal{D}$ and some $\mathbf{x}_i$. We then solve:

$$\text{SOP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & q_1\left(\mathbf{x}_i\right) \leq \psi, \forall \mathbf{x}_i \in \mathcal{S}, \\ & q_2\left(\mathbf{x}_i\right) \leq \psi, \forall \mathbf{x}_i \in \mathcal{U}, \\ & q_3\left(\mathbf{x}_i\right) \leq \psi, \forall \mathbf{x}_i \in \mathcal{D}, \\ & \psi \in \mathbb{R} \end{cases} \tag{5.11}$$

where $q_k(\mathbf{x}), k \in 1,2,3$ are as defined in (5.10), and the data sets $\mathcal{S}$,$\mathcal{U}$, and $\mathcal{D}$ correspond to points sampled from the initial safe set $\mathcal{X}_I$, initial unsafe set $\mathcal{X}_U$, and state space $\mathcal{X}$, respectively.

Since SOP is a linear program in the decision variable ψ, a feasible solution can be obtained, denoted as ψ^*. The following theorem establishes conditions ensuring that the SNCBF $B(\mathbf{x})$ obtained via (5.11) provides a valid solution to Problem 5.1.

Theorem 5.4.2. *Consider a continuous time stochastic control system (5.1), and initial safe and unsafe sets $\mathcal{X}_I \subseteq \mathcal{X}_S \subseteq \mathcal{X}$ and $\mathcal{X}_U \subseteq \mathcal{X} \backslash \mathcal{X}_S$, respectively. Let $B(\mathbf{x})$ be the SNCBF with trainable parameters θ. Suppose condition 1) in Assumption 5.3 holds so that the functions $q_k(\mathbf{x}), k \in \{1,2,3\}$ in equation (5.10) are Lipschitz continuous. Given $\bar{\epsilon}$, suppose we sample N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ such that*

for any point $\mathbf{x}$ *in the state space, there exists a sampled point* $\mathbf{x}_i$ *satisfying* $\|\mathbf{x} - \mathbf{x}_i\| \leq \bar{\epsilon}$. *For the SOP* (5.11) *constructed by utilizing* $\mathbf{x}_1, \ldots, \mathbf{x}_N$ *such that* $\mathcal{D}$ *is covered by balls centered at the* $\mathbf{x}_i$*'s with radii* $\bar{\epsilon}$, *let* ψ^* *be the optimal value. Then* $B(\mathbf{x})$ *is a valid* SNCBF, *i.e., it solves Problem* 5.1, *if the following condition holds:*

$$L_{\max}\bar{\epsilon} + \psi^* \leq 0, \tag{5.12}$$

where $L_{\max}$ *is the maximum of the Lipschitz constants of* $q_k(\mathbf{x}), k \in \{1, 2, 3\}$ *in* (5.10).

Proof. For any $\mathbf{x}$ and any $k \in \{1, 2, 3\}$, we know that:

$$\begin{aligned} q_k(\mathbf{x}) &= q_k(\mathbf{x}) - q_k(\mathbf{x}_i) + q_k(\mathbf{x}_i) \\ &\leq L_k \|\mathbf{x} - \mathbf{x}_i\| + \psi^* \\ &\leq L_k\bar{\epsilon} + \psi^* \leq L_{\max}\bar{\epsilon} + \psi^* \leq 0. \end{aligned}$$

Hence, if $q_k(\mathbf{x}), k \in \{1, 2, 3\}$ satisfies condition (5.12), then the $B(\mathbf{x})$ is a valid SCBF, satisfying conditions (5.8). □

Theorem 5.4.2 reformulates Problem 5.1 as follows. Given the data sets $\mathcal{S}$,$\mathcal{U}$ and $\mathcal{D}$, devise an algorithm that synthesizes SNCBF $B(x)$ such that it satisfies the conditions required in SOP (5.11) and ψ^* satisfies condition (5.12).

We next present the method to synthesize a smooth SNCBF. Let $\mathbf{e}_i$ denote a one-hot vector, e.g., $[\mathbf{e}_i]_i = 1$ and $[\mathbf{e}_i]_j = 0$ for $j \neq i$. We define a diagonal weighting matrix as follows.

$$\Omega := \left\{ \Omega \in \mathbb{R}^{n\times n} \mid \Omega = \sum_{i=1}^{n} \omega_{ii}\mathbf{e}_i\mathbf{e}_i^T, \omega_{ii} \geq 0 \right\}.$$

For an NN with one hidden layer, we have $\Omega \in \mathcal{D}_{n_x}$. By [117, Section III], the NN is Lipschitz continuous with coefficient L if there exists a nonnegative diagonal matrix Ω such that the matrix $M(\theta, \Omega)$ defined by

$$M(\cdot) = \begin{bmatrix} L^2\mathbf{I} + 2\underline{\varphi'}\overline{\varphi'}\mathbf{W}_1^T\Omega\mathbf{W}_1 & -(\underline{\varphi'} + \overline{\varphi'})\mathbf{W}_1^T\Omega & 0 \\ -(\underline{\varphi'} + \overline{\varphi'})\Omega\mathbf{W}_1 & 2\Omega & -\mathbf{W}_2 \\ 0 & -\mathbf{W}_2 & \mathbf{I} \end{bmatrix}$$

is positive semidefinite.

Our scenario necessitates ensuring not only the Lipschitz boundedness of the SNCBF $B(\mathbf{x})$, but also of $\frac{\partial B}{\partial \mathbf{x}}$ and $\sigma^\intercal \frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma$. Therefore, we must explore the relation-

ship between the network weights and the matrix M to guarantee the boundedness of the aforementioned terms. To address this issue, we introduce the following theorem which presents a method to ensure the L-Lipschitz continuity of the SNCBF $B(\mathbf{x})$, its derivative, and the Hessian terms in (5.10).

Theorem 5.4.3. *Suppose Assumption 5.3 holds for an SNCBF $B(\mathbf{x})$. The certificate for L-Lipschitz continuity of $\frac{\partial B}{\partial \mathbf{x}}$ which is the derivative of the NN is given by $M_{\hat{\varphi}}(\hat{\theta}, \Omega) \succeq 0$, where $\hat{\varphi} = \varphi'$ and $\hat{\theta} = (\mathbf{W}_1, \hat{\mathbf{W}}_2)$, $\hat{\mathbf{W}}_2$ is defined as:*

$$\hat{\mathbf{W}}_2 = \mathbf{W}_1^T \mathsf{diag}(\mathbf{W}_2). \tag{5.13}$$

Additionally, the certificate for the L-Lipschitz continuity of $\mathsf{tr}(\sigma^T \frac{\partial^2 \mathbf{y}}{\partial x^2} \sigma)$ is expressed as $M_{\bar{\varphi}}(\bar{\theta}, \Omega) \succeq 0$, where $\bar{\varphi} = \varphi''$ and $\bar{\theta} = (\mathbf{W}_1, \bar{\mathbf{W}}_2)$ and $\bar{\mathbf{W}}_2$ is defined as:

$$\bar{\mathbf{W}}_2 = \begin{bmatrix} \sum_{j=0}^{r} \sigma_j^2 \mathbf{W}_2^{j1} \mathbf{W}_1^{1j} & \dots & \sum_{j=0}^{r} \sigma_j^2 \mathbf{W}_2^{jn} \mathbf{W}_1^{pj} \end{bmatrix}. \tag{5.14}$$

Proof. Consider an NN with p neurons in a hidden layer. The dimension of the input $(\mathbf{x})$ is $n_x \times 1$, the dimension of pre-final weight $(\mathbf{W}_1)$ is $p \times n_x$, the dimension of pre-final bias $(\mathbf{r}_1)$ is $p \times 1$ and the dimension of final weight $(\mathbf{W}_2)$ is $1 \times p$. Let us start by differentiating the NN:

$$\begin{aligned} \mathbf{y} &= \mathbf{W}_2 \varphi(\mathbf{W}_1 x + \mathbf{r}_1) \\ \frac{\partial \mathbf{y}}{\partial \mathbf{x}} &= \mathbf{W}_2 \mathsf{diag}(\varphi') \mathbf{W}_1 \\ &\quad (\text{Here, } \varphi' = \varphi'(\mathbf{W}_1 \mathbf{x} + \mathbf{r}_1)). \end{aligned}$$

The dimension of $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ is $1 \times n_x$, therefore, its transpose will have the dimension of $n_x \times 1$.

$$\begin{aligned} (\frac{\partial \mathbf{y}}{\partial \mathbf{x}})^T &= (\mathbf{W}_2 \mathsf{diag}(\varphi') \mathbf{W}_1)^T \\ &= ((\varphi')^T \mathsf{diag}(\mathbf{W}_2) \mathbf{W}_1)^T \\ &= \underbrace{\mathbf{W}_1^T \mathsf{diag}(\mathbf{W}_2)}_{\hat{\theta}_l} \varphi'(\mathbf{W}_1 x + \mathbf{r}_1). \end{aligned}$$

The term derivative term $(\frac{\partial \mathbf{y}}{\partial \mathbf{x}})$ is equivalent to an NN with with activation $\hat{\varphi} = \varphi_l'$ and weight parameters $\hat{\theta} = (\mathbf{W}_1, \hat{\mathbf{W}}_2)$.
Therefore, the certificate for L-Lipschitz continuity of the derivative term $(\frac{\partial \mathbf{y}}{\partial \mathbf{x}})$ is given by $M_{\hat{\varphi}}(\hat{\theta}, \Omega) \succeq 0$.
Now, let us calculate $\mathsf{tr}(\sigma^T \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x}^2} \sigma)$, where σ is $n_x \times n_x$ diagonal matrix.

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \hat{\mathbf{W}}_2 \hat{\varphi}(\mathbf{W}_1 x + \mathbf{r}_1)$$
$$\sigma^T \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x}^2} \sigma = \sigma^T (\hat{\mathbf{W}}_2 \mathsf{diag}(\hat{\varphi}') \mathbf{W}_1) \sigma$$
$$\mathsf{tr}(\sigma^T \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x}^2} \sigma) = \mathsf{tr}(\sigma^T (\hat{\mathbf{W}}_2 \mathsf{diag}(\hat{\varphi}') \mathbf{W}_1) \sigma)$$

$$= \underbrace{\left[\sum_{j=0}^{r} \sigma_j^2 \hat{\mathbf{W}}_2^{j1} \mathbf{W}_1^{1j} \dots \sum_{j=0}^{r} \sigma_j^2 \hat{\mathbf{W}}_2^{jp} \mathbf{W}_1^{pj}\right]}_{\bar{\mathbf{W}}_2} \varphi''(\mathbf{W}_1 \mathbf{x} + \mathbf{r}_1).$$

The term $\mathsf{tr}(\sigma^T \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x}^2} \sigma)$ is equivalent to a feedforward NN with activation $\bar{\varphi} = \varphi''$ and weight parameters $\bar{\theta} = (\mathbf{W}_1, \bar{\mathbf{W}}_2)$ and $\hat{\mathbf{W}}_2$ is defined as:

$$\bar{\mathbf{W}}_2 = \begin{bmatrix} \sum_{j=0}^{n_x} \sigma_j^2 \hat{\mathbf{W}}_2^{j1} \mathbf{W}_1^{1j} & \dots & \sum_{j=0}^{n_x} \sigma_j^2 \hat{\mathbf{W}}_2^{jp} \mathbf{W}_1^{pj} \end{bmatrix}.$$

Therefore, the certificate for Lipschitz continuity of the $\mathsf{tr}(\sigma^T \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x}^2} \sigma)$ term is given by $M_{\bar{\varphi}}(\bar{\theta}, \Omega) \succeq 0$. □

To train the SNCBF to satisfy (5.8), we define a loss function with three terms matching the three constraints. The first penalizes $B(\mathbf{x}_i) < 0$ for $\mathbf{x}_i \in \mathcal{X}_I \subseteq \mathcal{X}_S$, ensuring non-negativity in the initial safe set. The second penalizes $B(\mathbf{x}_i) \geq -\delta$ for $\mathbf{x}_i \in \mathcal{X}_U$, ensuring the barrier is sufficiently negative in the unsafe set. The third introduces a slack variable ψ, allowing for small constraint violations, quantifies infeasibility.

$$\begin{aligned} \mathcal{L}(\theta) =& \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{S}} \max\left(0, (-B(\mathbf{x}_i))\, \mathbb{1}_{\mathcal{X}_I} - \psi\right), \\ &+ \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{U}} \max\left(0, (B(\mathbf{x}_i) + \delta)\, \mathbb{1}_{\mathcal{X}_U} - \psi\right), \\ &+ \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{D}} \max(0, -\frac{\partial B}{\partial \mathbf{x}} f(\mathbf{x}_i) - \frac{\partial B}{\partial \mathbf{x}} g(\mathbf{x}_i) u - \\ &\frac{1}{2} \mathsf{tr}\left(\sigma^\intercal \frac{\partial^2 B(\mathbf{x}_i)}{\partial \mathbf{x}^2} \sigma\right) - \alpha(B(\mathbf{x}_i)) - \psi). \end{aligned} \tag{5.15}$$

Let us consider a constrained optimization problem aiming to minimize loss $\mathcal{L}(\theta)$ in (5.15) subject to $M_j(\theta, \Omega) \succeq 0$ for $j = 1, 2, 3$, to ensure the Lipschitz continuity of the SNCBF $B(\mathbf{x})$, as well as its derivative and Hessian terms. By employing a log-determinant barrier function, we convert this into an unconstrained optimization problem:

$$\min_{\theta, \Omega} \mathcal{L}(f_\theta) + \mathcal{L}_M(\theta, \Omega),$$

where $\mathcal{L}_M(\theta, \Omega) = -\sum_{j=0}^{q} \rho_j \log\det(M_j(\theta, \Omega))$ and $\rho_j > 0$ are barrier parameters. Ensuring that the loss function $\mathcal{L}_M(\theta, \Omega) \leq 0$, guarantees that the linear matrix

inequalities $M_j(\theta, \Omega) \succeq 0, j = 1, 2, 3$ hold true. Let us consider the loss functions characterizing the satisfaction of Lipschitz bound as

$$\begin{aligned}\mathcal{L}_M(\theta, \Omega, \hat{\Omega}, \bar{\Omega}) = &-c_{l_1} \log\det(M_1(\theta, \Omega)) \\ &- c_{l_2} \log\det(M_2(\hat{\theta}, \hat{\Omega})) - c_{l_3} \log\det(M_3(\bar{\theta}, \bar{\Omega})),\end{aligned} \tag{5.16}$$

where $c_{l_1}, c_{l_2}, c_{l_3}$ are positive weight coefficients for the sub-loss functions, M_1, M_2, M_3 are the semi-definite matrices corresponding to the Lipschitz bounds L_h, L_{dh}, L_{d2h} respectively, $\Omega, \hat{\Omega}, \bar{\Omega}$ are trainable parameters and $\theta, \hat{\theta}, \bar{\theta}$ are the weights mentioned in Theorem 5.4.3.

Finally, let us consider the following loss function to satisfy validity condition (5.12):

$$\mathcal{L}_v(\psi) = \max\left(0, L_{\max}\bar{\epsilon} + \psi\right), \tag{5.17}$$

where $L_{\max}$ is maximum of the Lipschitz constants of $q_k(x), k \in \{1, 2, 3\}$ in (5.10), or $L_{\max} = \max\left(L_h, L_h + L_{dh}L_x + L_{d2h}\right)$ and $\bar{\epsilon}$ is our sampling density.

The training procedure begins by fixing all hyperparameters, including $\bar{\epsilon}, \mathcal{L}_h, \mathcal{L}_{dh}, \mathcal{L}_{d2h}, \omega_1, \omega_2, c_{l_1}, $ and the maximum number of epochs. The $\texttt{Learn}(\mathcal{L}, \cdot)$ function performs gradient-based optimization to minimize loss $\mathcal{L}$ with respect to its second argument (network parameters, Lipschitz bound parameters, or slack variables). We employ stochastic gradient descent (SGD) with learning rate λ for K iterations. The overall algorithm is summarized in Algorithm 4.

Algorithm 4 Training Formally Verified SNCBF

Require: Inputs: Data sets $\mathcal{S}, \mathcal{U}, \mathcal{D}$, stochastic dynamics f, g, σ, cover radius $\bar{\epsilon}$, Lipschitz bounds $L_h, L_{dh}, L_x, L_{d2h}$

Ensure: Output: Formally verified smooth SNCBF B_θ for deployment through the SNCBF-QP

Initialize$(\theta, \psi, \Omega, \Omega', \Omega'')$
$x_i \leftarrow sample(\mathcal{S}, \mathcal{U}, \mathcal{D})$
$L_{\max} \leftarrow L_h, L_{dh}, L_x, L_{d2h}$
while $\mathcal{L}_\theta > 0$ or $\mathcal{L}_M \not\leq 0$ or $\mathcal{L}_v > 0$ **do**
  $B \leftarrow \theta$
  $u_i \leftarrow$ SNCBF_QP$(B, f, g, \sigma, \mathbf{x}_i, \psi)$ ▷ From eq. (5.4)
  $\mathcal{L}_\theta \leftarrow (B, f, g, \sigma, \mathbf{x}_i, u_i, \psi)$ ▷ From eq. (5.15)
  $\theta \leftarrow Learn(\mathcal{L}_\theta, \theta)$
  $\mathcal{L}_M \leftarrow (\theta, \Omega, \Omega', \Omega'')$ ▷ From eq. (3.9)
  $\theta, \Omega, \Omega', \Omega'' \leftarrow Learn(\mathcal{L}_M)$
  $\mathcal{L}_v \leftarrow (L_{\max}, \psi)$ ▷ From eq. (3.10)
  $\psi \leftarrow Learn(\mathcal{L}_v)$
end while

We note that the verification-free synthesis approach requires one hidden layer (Assumption 5.3) and dense sampled training dataset. Section 5.4.2 relaxes both as-

sumptions by introducing VITL synthesis with SMT-based verification, which relaxes the dense sampling requirement.

5.4.2 Smooth SNCBF Verification and Synthesis

The previous approach is limited to single hidden layer NNs and requires dense sampling under Assumption 5.3. In this subsection, we consider an NN with multiple hidden layers. To handle the increased complexity, we employ a verification-in-the-loop (VITL) framework that iteratively refines the SNCBF parameters based on verification results. This approach relaxes Assumption 5.3, allowing for more expressive NN architectures while still ensuring safety. The safety guarantee of our smooth SNCBF relies on both the correctness of the super-0-level set, i.e., $\mathcal{D} \subseteq \mathcal{X}_S$ and the existence of the control signal u satisfying the SNCBF condition, given as follows.

Definition 5.4.4 (Valid Smooth SNCBFs)**.** The function $B(\mathbf{x})$ is called a valid smooth SNCBF if $B(\mathbf{x})$ satisfies the *Correctness* requirement of SNCBFs, i.e. $\forall \mathbf{x} \in \mathcal{D}$, and the *Feasibility* requirement of SNCBFs i.e. $\mathbf{x} \in \mathcal{X}_S$, and there exists a stochastic smooth NCBF control u satisfying the conditions of Proposition 5.3.4.

These correctness and feasibility conditions must hold for every state in the super-0-level-set $\mathbf{x} \in \mathcal{D}$. States are referred to as correctness or feasibility counterexamples if they violate the corresponding condition. Since existing VITL cannot ensure the elimination of all counterexamples [145, 198], it is essential to verify SNCBF.

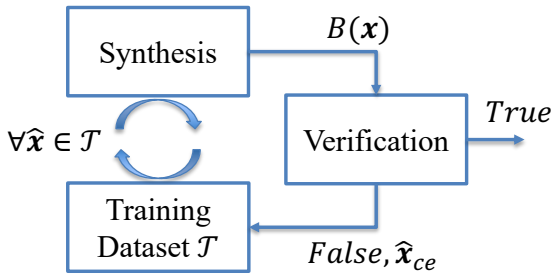


Figure 5.1: Workflow of the synthesis with verification in the loop.

As shown in Fig. 5.1, our proposed training framework consists of three components: a training dataset $\mathcal{T}$, a synthesis module, and a verification module. These elements form two loops. The inner loop learns a parameter θ for the synthesized $B(\mathbf{x})$ that satisfies the conditions of Definition 5.4.4 by minimizing a loss function over training data $\mathcal{T}$. The training dataset $\mathcal{T}$ is initialized through uniform sampling of $\mathcal{X}$. This loss function consists of a weighted sum of two components, structured as

an unconstrained optimization problem over θ.

$$\min_{\theta} \quad \lambda_f \mathcal{L}_f(\mathcal{T}) + \lambda_c \mathcal{L}_c(\mathcal{T}) \tag{5.18}$$

where λ_f and λ_c are non-negative coefficients. The term $\mathcal{L}_f(\mathcal{T})$ is the loss penalizing the violations of constraint (5.22a)-(5.22c) (feasibility condition of Definition 5.4.4), and the term $\mathcal{L}_c(\mathcal{T})$ penalizes the correctness counterexample with negative minima of (5.20) (correctness condition of Definition 5.4.4). For each sample $\hat{\mathbf{x}} \in \mathcal{T}$ the safe control signal $u(\mathbf{x})$ is calculated by SNCBF-QP. The loss $\mathcal{L}_f$ enforces the satisfaction of the constraint by inserting a positive relaxation term r in the constraint and minimizing r with a large penalization in the objective function. We have the loss $\mathcal{L}_f$ defined as $\mathcal{L}_f = ||u(\mathbf{x}) - u_{ref}(\mathbf{x})||^2 + r$. The loss term $\mathcal{L}_c$ is defined as

$$\mathcal{L}_c = \frac{1}{N} \sum_{\hat{\mathbf{x}}_i \in \mathcal{X}_I} \max\left(0, (\epsilon - B(\hat{\mathbf{x}}_i))\, \mathbb{1}_{\mathcal{X}_s}\right) + \frac{1}{N} \sum_{\hat{\mathbf{x}}_j \in \mathcal{X}_U} \max\left(0, (B(\hat{\mathbf{x}}_j) + \epsilon)\, \mathbb{1}_{\mathcal{X}_U}\right). \tag{5.19}$$

The outer loop validates a given SNCBF $B(\mathbf{x})$ by searching for counterexamples $\hat{\mathbf{x}}_{ce}$ and updates the training dataset as $\mathcal{T} \cup \{\hat{\mathbf{x}}_{ce}\}$. The framework penalizes constraint violations for all $\hat{\mathbf{x}} \in \mathcal{T}$ and extends these guarantees from $\mathcal{T}$ to the entire state space $\mathcal{X}$ through formal verification. Utilizing this framework results in the verification of the synthesized SNCBF while also generating counterexamples to augment the training dataset.

In what follows, we present the verification of the smooth SNCBF. We first verify the correctness condition in Definition 5.4.4. What we want to show is that the super-0-level set of the SNCBF is contained by the safe region $\mathcal{D} \subseteq \mathcal{X}_S$. The correctness counterexample refers to states $\mathbf{x} \in \mathcal{D} \subseteq \mathcal{X}_S$ with $h(\mathbf{x}) < 0$. Specifically, we check if $\mathcal{D} \cap \mathcal{X}_U = \emptyset$. It suffices to solve the nonlinear program

$$\begin{aligned} &\min_{\mathbf{x}} \quad h(\mathbf{x}) \\ &\text{s.t.} \quad\ B(\mathbf{x}) \geq 0. \end{aligned} \tag{5.20}$$

If there exists $\mathbf{x}^*$ such that $h(\mathbf{x}^*) < 0$ while $B(\mathbf{x}^*) = 0$, then $\mathbf{x}^*$ is a correctness counterexample.

Next, we verify the feasibility condition in Definition 5.4.4. Let $\lambda(\mathbf{x}_t)$ and $\xi(\mathbf{x}_t)$ be as follows.

$$\lambda(\mathbf{x}_t) := \frac{\partial B}{\partial \mathbf{x}_t} g(\mathbf{x}_t)$$

$$\xi(\mathbf{x}_t) := \frac{\partial B}{\partial \mathbf{x}_t} f(\mathbf{x}_t) + \frac{1}{2}\mathbf{tr}\left(\sigma^T \frac{\partial^2 B}{\partial \mathbf{x}_t^2}\sigma\right) - \alpha(B(\mathbf{x}_t)).$$

Rearranging the terms in (5.8), we have the affine constraints given as

$$\lambda(\mathbf{x}_t)u_t \leq \xi(\mathbf{x}_t), \tag{5.21}$$

The feasibility counterexample refers to the state $\mathbf{x}$ such that there does not exist a u satisfying condition (5.21).

We next consider the case with control input constraints $u \in \mathcal{U} := \{u : Au \leq \mathbf{b}\}$. We present $\Lambda(\mathbf{x}_t)$, $\Xi(\mathbf{x}_t)$ and corresponding verification formulation.

Proposition 5.4.5. *Let $\Lambda(\mathbf{x}_t)$ and $\Xi(\mathbf{x}_t)$ be as follows.*

$$\Lambda(\mathbf{x}_t) := \begin{bmatrix} -\lambda(\mathbf{x}_t) \\ A \end{bmatrix}; \qquad \Xi(\mathbf{x}_t) := \begin{bmatrix} \xi(\mathbf{x}_t) \\ \mathbf{b} \end{bmatrix}$$

There always exists a u satisfying (5.21) if and only if there is no $\mathbf{x} \in \mathcal{D}$ and $\mathbf{y} \in \mathbb{R}^{n_u+1}$ such that

$$[\mathbf{y}]_i \geq 0, \ \forall i \in \{1, \ldots, n_u + 1\} \tag{5.22a}$$
$$-\mathbf{y}^T \Xi(\mathbf{x}) < 0 \tag{5.22b}$$
$$\left[\mathbf{y}^T \Lambda(\mathbf{x})\right]_i = 0, \ \forall i \in \{1, \ldots, n_u + 1\} \tag{5.22c}$$

Proof. The proof is to show that there does not exist $\mathbf{x}$ such that, for any $u \in \mathcal{U}$, (5.21) fails to hold. By Farkas's Lemma, the existence of control input u satisfying (5.21) is equivalent to the non-existence of $\mathbf{y}$ satisfying (5.22a)–(5.22c), completing the proof. □

The NN is a polynomial if all the activation functions are polynomials, e.g., Hermite Polynomial Activation Functions [105]. The following lemma shows that given a polynomial SNCBF, the non-existence of $\mathbf{x}$ and $\mathbf{y}$ can be proved through Positivstellensatz.

Lemma 5.2. *There is no feasibility counterexample $\mathbf{x} \in \mathcal{D}$ if and only if there exist polynomials $\rho_1^{\mathbf{y}}(\mathbf{x}) \ldots \rho_{n_u+1}^{\mathbf{y}}(\mathbf{x})$, sum-of-squares polynomials $q_P(\mathbf{x})$, integers $\tau = n_u + 3$, r_1 such that*

$$\phi(\mathbf{x}, \mathbf{y}) + \chi(\mathbf{x}, \mathbf{y}) + \Gamma(\mathbf{x}, \mathbf{y}) = 0, \tag{5.23}$$

and

$$\begin{aligned}\phi(\mathbf{x},\mathbf{y}) &= \sum_{K\subseteq\{1,\dots,\tau\}} p_i(\mathbf{x}) \prod_{i\in K} \phi_i(\mathbf{x},\mathbf{y}) \\ \chi(\mathbf{x},\mathbf{y}) &= \phi_1(\mathbf{x},\mathbf{y})^{2r_1} \\ \Gamma(\mathbf{x},\mathbf{y}) &= \sum_{i=1}^{n_u} \rho_i^{\mathbf{y}} \left[\mathbf{y}^T \Lambda(\mathbf{x})\right]_i,\end{aligned}$$

where $\phi_1(\mathbf{x}) = -\mathbf{y}^T\Xi(\mathbf{x})$, $\phi_2(\mathbf{x}) = B(\mathbf{x})$, *and* $\phi_{3,\dots,n_u+3}(\mathbf{x},\mathbf{y}) = y_i$.

Proof. By Proposition 5.4.5, we have that there is no feasibility counterexample iff there exists no $\mathbf{x} \in \mathcal{D}$ and $\mathbf{y}$ such that (5.22a)–(5.22c) hold. The condition $\mathbf{y}^T\Xi(\mathbf{x}) < 0$ is equivalent to $-\mathbf{y}^T\Xi(\mathbf{x}) \geq 0$ and $\mathbf{y}^T\Xi(\mathbf{x}) \neq 0$. The result then follows from the Positivstellensatz □

If the SNCBF has non-polynomial activation functions, the conditions of Proposition 5.4.5 can be verified by solving the nonlinear program

$$\begin{aligned} \min_{\mathbf{x},y} \quad & \mathbf{y}^T\Xi(\mathbf{x}) \\ \text{s.t.} \quad & B(\mathbf{x}) \geq 0 \\ & [\mathbf{y}]_i \geq 0, \ \forall i \in \{1,\dots,n_u+1\} \\ & \left[\mathbf{y}^T\Lambda(\mathbf{x})\right]_i = 0, \ \forall i \in \{1,\dots,n_u+1\} \end{aligned} \tag{5.24}$$

and checking whether the optimal value is nonnegative (safe) or negative (unsafe).

The following corollary describes the special case where there are no constraints on the control, i.e., $\mathcal{U} = \mathbb{R}^{n_u}$.

Corollary 5.4.6. *There exists* $u \in \mathbb{R}^{n_u}$ *satisfying*

$$-\lambda(\mathbf{x}_t)u \leq \xi(\mathbf{x}_t)$$

for all $\mathbf{x} \in \mathcal{D}$ *if and only if there is no* $\mathbf{x}$ *such that* $\lambda(\mathbf{x}) = \mathbf{0}$, *and* $\xi(\mathbf{x}) > 0$.

Proof. When the vector $\frac{\partial B}{\partial x} g(\mathbf{x}) \neq \mathbf{0}$, there is always $u \in \mathbb{R}^{n_u}$ that satisfies (5.21). When the vector $\frac{\partial B}{\partial x} g(\mathbf{x}) = \mathbf{0}$, u does not affect the inequality, which is equivalent to having $u = \mathbf{0}$. In this case, $\Xi(\mathbf{x}_t) > 0$ ensures (5.21) hold. □

Under the condition of Corollary 5.4.6, the nonlinear program (5.24) reduces to

$$\begin{aligned} \min_{\mathbf{x}} \quad & \xi(\mathbf{x}) \\ \text{s.t.} \quad & B(\mathbf{x}) \geq 0 \\ & [\lambda(\mathbf{x})]_i = 0, \ \forall i \in \{1, \ldots, n_u\} \end{aligned} \tag{5.25}$$

The condition of Corollary 5.4.6 can be verified by checking whether the optimal value is nonnegative (safe) or negative (unsafe). Nonlinear programs (5.20), (5.24), and (5.25) can be solved by SMT solvers with numerical completeness guarantees.

5.5 Rectified Linear Unit Stochastic Control Barrier Functions

In this section, we investigate the SNCBF with non-smooth activation functions, specifically, the Rectified Linear Unit (ReLU) function $ReLU(x) = \max\{x, 0\}$. ReLU, one of the most popular activation functions, is not differentiable at point 0. This makes the approach of the previous section, where we assumed the activation functions were twice-differentiable everywhere, not directly applicable. To address this issue, we utilize Tanaka's formula, derive the safety condition, and propose efficient verification algorithms for VITL frameworks.

Beyond demonstrating that non-smooth activations are admissible, ReLU SNCBFs are motivated by three practical considerations. First, ReLU networks train faster and more reliably than smooth-activation networks of comparable size and dominate modern practice; certifying them directly avoids imposing smooth activations solely for verifiability, which typically degrades fit. Second, piecewise-linearity enables *exact* enumeration-based verification—the network partitions $\mathcal{X}$ into finitely many polytopes on each of which $\dot{B}$ is affine—whereas verifying twice-differentiable activations relies on conservative nonlinear-programming relaxations (cf. Remark 5.3.5). Third, the enumeration viewpoint relaxes the dense-sampling requirements of earlier chapters and, combined with VITL, permits deeper architectures than the single-hidden-layer smooth case.

Problem 5.2. *Given a continuous-time stochastic control system defined as (5.1), a desired confidence level η, state space $\mathcal{X}$, initial safe and unsafe sets $\mathcal{X}_s$ and $\mathcal{X}_U$, respectively, devise an algorithm that synthesizes a stochastic neural CBF* (SNCBF) *$B_\theta(\mathbf{x})$ with ReLU activation functions. In particular, the SNCBF*

must satisfy the following worst-case probability guarantee:

$$\mathbb{P}\left[\mathbf{x}_t \in \mathcal{X}_S, 0 \forall t \in [0,T] \mid \mathbf{x}_0 \in int(\mathcal{D})\right] \geq 1-\eta.$$

Our approach to this problem mirrors our approach for the smooth case: (i) deriving the safety condition of ReLU SNCBFs, (ii) verification of ReLU SNCBFs, and (iii) synthesis with verification in the loop.

5.5.1 Single-Hidden-Layer ReLU Stochastic NCBF

Since the ReLU neural network is not differentiable everywhere, the approach of the previous section is not directly applicable. Tanaka's formula provides a general form for creating NCBFs for stochastic systems with non-smooth activation functions by explicitly linking the local time of a semimartingale process to its behavior. Let $B : \mathbb{R}^n \to \mathbb{R}$ be a single-layer feedforward NN with ReLU activation function. Define $z_{1jt} = W_{1j}^T \mathbf{x}_t + r_{1j}$ and let $\mathcal{S}(t) = \{j : W_{1j}^T \mathbf{x}_t + r_{1j} > 0\}$, i.e., the set of neurons whose pre-activation inputs are positive at time t.

By Tanaka's formula, we have the random process $B(\mathbf{x}_t)$ satisfies

$$B(\mathbf{x}_t) = \sum_{j=1}^{M} [W_{2j}(\{z_{1j0}\}_+ + \int_0^t \mathbb{1}(z_{1js} > 0) dz_{1js})] + \frac{1}{2}\sum_{j=1}^{M} W_{2j}(|z_{1jt}| - |z_{1j0}| - \int_0^t \text{sgn}(z_{1js}) dz_{1js}) + r_2. \quad (5.26)$$

Let $\mathcal{D} := \{\mathbf{x} : B(\mathbf{x}) \geq 0\}$. We next derive a stochastic process that lower bounds $B(\mathbf{x}_t)$.

Lemma 5.3. *Suppose that there exists a scalar R_j such that $|z_j| \leq R_j$ for all $j \in \{1, \ldots, M\}$ whenever $B(\mathbf{x}_t) \geq 0$. Let $\tilde{B}(\mathbf{x}_t)$ be defined as follows.*

$$\tilde{B}(\mathbf{x}_t) := \sum_{j=1}^{M} [W_{2j}(\{z_{1j0}\}_+ + \int_0^t \mathbb{1}(z_{1js} > 0)\ dz_{1js})] + \frac{1}{2}\sum_{j=1}^{M} W_{2j}(-|z_{1j0}| - \int_0^t sgn(z_{1js})\ dz_{1js}) - \frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2 + r_2 \quad (5.27)$$

Then we have $\tilde{B}(\mathbf{x}_t) \leq B(\mathbf{x}_t)$ *for all* $\mathbf{x}_t \in \mathcal{D}$.

Proof. Given that there exists a scalar R_j such that $|z_{1jt}| \leq R_j$ for all $j \in \{1, \ldots, M\}$ whenever $B(\mathbf{x}_t) \geq 0$, we have $|z_{1jt}| \geq \frac{1}{R_j}|z_{1jt}|^2$ for all t when $B(x_t) \geq 0$. When $W_{2j} < 0$ and $W_{2j} \geq 0$, we have $\sum_{j=1}^{M} W_{2j}|z_{1jt}| \geq -\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2$. Hence, we have $\tilde{B}(\mathbf{x}_t) \leq B(\mathbf{x}_t)$ for all $\mathbf{x}_t \in \mathcal{D}$. □

For simplicity, we let $\zeta(\mathbf{x}_s) = 2\mathbf{x}_s^T W_{1j} W_{1j}^T + 2r_{1j} W_{1j}^T$. The expected instantaneous rate of change of $\tilde{B}_t := \tilde{B}(\mathbf{x}_t)$ is described by the Infinitesimal Generator $\mathcal{A}\tilde{B}_t$ defined as follows.

Lemma 5.4. *The infinitesimal generator of* $\tilde{B}_t$ *is*

$$\mathcal{A}\tilde{B}_t = \beta_D(\mathbf{x}_t, u_t) \tag{5.28}$$

where

$$\beta_D(\cdot) = \frac{1}{2}\sum_{j=1}^{M} W_{2j}\left(\mathbb{1}(z_{1js} > 0) - \frac{1}{2} sgn(z_{1js})\right) \tag{5.29}$$

$$\cdot W_{1j}^T(f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) \tag{5.30}$$

$$-\frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j}\Big[\zeta(\mathbf{x}_s)(f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) \tag{5.31}$$

$$+\frac{1}{2}\mathbf{tr}(\sigma_s^T W_{1j} W_{1j}^T \sigma_s)\Big].$$

Proof. Consider $\tilde{B}(x_t)$ defined by

$$\tilde{B}(\mathbf{x}_t) = \tilde{B}_0 - \frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2 + \sum_{j=1}^{M} W_{2j}\int_0^t (\mathbb{1}(z_{1js} > 0) - \frac{1}{2}\mathrm{sgn}(z_{1js}))dz_{1js},$$

where $\tilde{B}_0 = \sum_{j=1}^{M}(W_{2j}(\{z_{1j0}\}_+ - \frac{1}{2}|z_{1j0}|) + r_2$. By Ito's lemma [81], we have

$$dz_{1jt} = W_{1j}^T d\mathbf{x}_t = W_{1j}^T(f(\mathbf{x}_t) + g(\mathbf{x}_t)u_t dt + W_{1j}^T \sigma_t dW_t.$$

$$z_{1jt}^2 = z_{1j0}^2 + \int_0^t \Big[\zeta(\mathbf{x}_s)(f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) + \frac{1}{2}\mathbf{tr}(\sigma_s^T W_{1j} W_{1j}^T \sigma_s)\Big]\, ds + \int_0^t \zeta(\mathbf{x}_s)\sigma_s\, dW_s.$$

Substituting into the formula for $\tilde{B}(x_t)$ gives

$$\begin{aligned}
\tilde{B}(\mathbf{x}_t) = \tilde{B}_0 + \sum_{j=1}^{M} W_{2j} \Bigg[& \int_0^t \left(\mathbb{1}(z_{1js} > 0) - \frac{1}{2}\mathrm{sgn}(z_{1js}) \right) && (5.32)\\
& \cdot W_{1j}^T (f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s)\, ds && (5.33)\\
& + \int_0^t (\mathbb{1}(z_{1js} > 0) - \frac{1}{2}\mathrm{sgn}(z_{1js})) W_{1j}^T \sigma_s\, dW_s \Bigg] && (5.34)\\
- \frac{1}{2} \sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} \Bigg[& \int_0^t (\zeta(\mathbf{x}_s)(f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) && (5.35)\\
& + \frac{1}{2}\mathbf{tr}(\sigma_s^T W_{1j} W_{1j}^T \sigma_s))ds + \int_0^t \zeta(\mathbf{x}_s)\sigma_s\, dW_s \Bigg].
\end{aligned}$$

This gives the drift term

$$\begin{aligned}
\beta_D(\mathbf{x}_s, u_s) = \sum_{j=1}^{M} W_{2j} & \left(\mathbb{1}(z_{1js} > 0) - \frac{1}{2}\mathrm{sgn}(z_{1js}) \right) && (5.36)\\
& \cdot W_{1j}^T (f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) && (5.37)\\
- \frac{1}{2} \sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} & \Big[\zeta(\mathbf{x}_s)(f(\mathbf{x}_s) + g(\mathbf{x}_s)u_s) && (5.38)\\
& + \frac{1}{2}\mathbf{tr}(\sigma_s^T W_{1j} W_{1j}^T \sigma_s) \Big].
\end{aligned}$$

□

By (5.26) and (5.27), we have

$$\tilde{B}(\mathbf{x}_t) = B(\mathbf{x}_t) - \frac{1}{2}\sum_{j=1}^{M} W_{2j}|z_{1jt}| - \frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2.$$

We next define the safe control constraint based on $\tilde{B}(\mathbf{x}_t)$ as follows.

Definition 5.5.1. A control signal set $\mathcal{U}_z := \{u_t : t \in [0, T]\}$ is a ReLU SNCBF control if there exists $k > 0$ and $R_j > 0$ for $j \in \{1, \ldots, M\}$, such that the

following conditions are satisfied at each time t:

$$\mathcal{A}\tilde{B}(\mathbf{x}_t) \geq -k\tilde{B}(\mathbf{x}_t). \tag{5.39}$$

Let $\tilde{\mathcal{D}} := \{\mathbf{x} : \tilde{B}(\mathbf{x}) \geq 0\}$. The following result shows the worst-case probability guarantee when applying stochastic ReLU NCBF control.

Theorem 5.5.2. *Suppose Assumption 5.1 holds and Assumption 5.2 holds for $\tilde{B}(\mathbf{x})$, i.e., $\tilde{B}(\mathbf{x})$ is in the domain of the operator $\mathcal{D}(\mathcal{A})$. Further suppose there exists a scalar R_j such that $|z_j| \leq R_j$ for all $j \in \{1, \ldots, M\}$ whenever $B(\mathbf{x}) \geq 0$. Let $\mathcal{U}_z$ be a ReLU SNCBF control set in Definition 5.5.1, $\mathbf{x}_0 \in \tilde{\mathcal{D}} \cap \mathcal{D}$ and $c = \sup_{x \in \tilde{\mathcal{D}}} \tilde{B}(\mathbf{x})$. By choosing $u_t \in \mathcal{U}_z$ for all time t, we have the following worst-case probability estimation:*

$$\mathbb{P}\left[\mathbf{x}_t \in int(\mathcal{D}), t \in [0, T]\right] \geq \left(\frac{\tilde{B}(\mathbf{x}_0)}{c}\right) e^{-cT}.$$

Proof. Given $\mathbf{x}_0 \in \tilde{\mathcal{D}} \cap \mathcal{D}$ and $u \in \mathcal{U}_z$, by Proposition 5.3.4, we have $\mathbf{x}_t \in int(\tilde{\mathcal{D}})$ with probability

$$\mathbb{P}\left[\mathbf{x}_t \in int(\tilde{\mathcal{D}}), t \in [0, T]\right] \geq \left(\frac{\tilde{B}(\mathbf{x}_0)}{c}\right) e^{-cT}.$$

By Lemma 5.3, we have $\tilde{B}(\mathbf{x}_t) \leq B(\mathbf{x}_t)$ for all $\mathbf{x}_t \in \mathcal{D}$.

$$\mathbb{P}\left[\mathbf{x}_t \in int(\mathcal{D}), t \in [0, T]\right] \geq \mathbb{P}\left[\mathbf{x}_t \in int(\tilde{\mathcal{D}}), t \in [0, T]\right]$$

Since $\mathcal{D} \subseteq \mathcal{X}_S$, we have

$$\mathbb{P}\left[\mathbf{x}_t \in int(\mathcal{D}), t \in [0, T]\right] \geq \mathbb{P}\left[\mathbf{x}_t \in int(\mathcal{D}), t \in [0, T]\right].$$

□

5.5.2 ReLU SNCBF Verification and Synthesis

The safety guarantee derived in Section 5.5.1 is dependent on if the given ReLU-SNCBF $\tilde{B}$ is a valid ReLU-SNCBF. In what follows, we first present the conditions of a valid ReLU-SNCBF and then present the verification approach.

Definition 5.5.3 (Valid ReLU SNCBFs)**.** The function $B(\mathbf{x})$ is a valid ReLU SNCBF if $\forall \mathbf{x} \in \mathcal{D}$ (*Correctness*) $\mathbf{x} \in \mathcal{X}_S$ and (*Feasibility*) there exists control u satisfying the ReLU SNCBF $\tilde{B}(\mathbf{x})$ conditions (5.39).

The correctness and feasibility verification rely on iterating all $\mathcal{S} \in \mathbf{Q}$, where $\mathbf{Q}$ is the set of all activated sets in our NN. Next, we present the enumeration of activated sets $\mathcal{S}$.

Enumeration

Let $\mathcal{S}(\mathbf{x})$ denote the activated set $\mathcal{S}$ determined by $\mathbf{x}$. Let $\triangle$ denote the symmetric difference between two sets. The activation flip j-th neuron is denoted as $\mathcal{S} \triangle \{j\}$, e.g. if $j \in \mathcal{S}$, then j is excluded by operation $\triangle$ from the set $\mathcal{S}$. Given a state $\mathbf{x}_0$, the following linear program checks the existence of a state $\mathbf{x}$ such that $B(\mathbf{x}) \geq 0$.

$$\text{SuperLP}(\mathcal{S}(\mathbf{x}_0)) = \begin{cases} \text{find} & \mathbf{x} \\ \text{s.t.} & \overline{W}(\{\mathcal{S}(\mathbf{x}_0)\})^T \mathbf{x} + \overline{r}(\{\mathcal{S}(\mathbf{x}_0)\}) \geq 0 \\ & \mathbf{x} \in \mathcal{X}(\mathcal{S}(\mathbf{x}_0)). \end{cases}$$

The Feasibility of SuperLP means the hyperplane $\mathcal{X}(\mathcal{S}(\mathbf{x}_0))$ contains the super-0-level-set of $B(\mathbf{x})$.

The enumeration process conducts its search in a breadth-first manner to determine if a neighboring hyperplane, with a flip in its j-th neuron, contains $\mathbf{x}$ such that $B(\mathbf{x}) \geq 0$. Enumeration solves a linear program, referred to as the unstable neuron linear program of $\text{USLP}(\mathcal{S}, j)$. This linear program checks the existence of a state $\mathbf{x} \in \mathcal{X}(\mathcal{S}) \cap \{\mathbf{x} : W_{ij}\mathbf{x} + r_{ij} = 0\}$ that satisfies $\overline{W}(\mathcal{S})\mathbf{x} + \overline{r}(\mathcal{S}) \geq 0$. The unstable neuron linear program is defined as follows.

$$\text{USLP}(\mathcal{S}, j) = \begin{cases} \text{find} & \mathbf{x} \\ \text{s.t.} & \overline{W}(\{\mathcal{S}\})^T \mathbf{x} + \overline{r}(\{\mathcal{S}\}) \geq 0 \\ & W_{ij}(\{\mathcal{S}\})^T \mathbf{x} + r_{ij}(\{\mathcal{S}\}) = 0 \\ & \mathbf{x} \in \mathcal{X}(\mathcal{S}). \end{cases}$$

The enumeration process described above is summarized in Algorithm 5.

Proposition 5.5.4. *Let* $\mathbf{Q}$ *denote the output of Algorithm 5. Then the super-0-level set* $\mathcal{D}$ *satisfies* $\mathcal{D} \subseteq \bigcup_{\mathcal{S} \in \mathbf{Q}} \mathcal{X}(\{\mathbf{S}\})$.

Algorithm 5 Enumerate Activated Sets

Require: Inputs: Candidate ReLU SNCBF B, initial state $\mathbf{x}_0 \in \text{int}(\mathcal{D})$, initial activation set $\mathcal{S}_0 = \mathcal{S}(\mathbf{x}_0)$

Ensure: Output: Activation sets $\mathbf{Q}$ intersecting the super-zero-level set

procedure ENUMERATION($\mathcal{S}_0$)
  Initialize queue $\mathcal{Q}$ with initial activation set $\mathcal{S}_0$
  Initialize sets $\mathcal{S}$ with $\mathcal{S}_0$
  while queue $\mathcal{Q}$ is not empty **do**
    Dequeue $\mathcal{S}$ from $\mathcal{Q}$ ▷ Pop the first activated set
    if SuperLP($\mathcal{S}$) is feasible **then**
      if $\mathcal{S} \notin \mathbf{Q}$ **then** ▷ If $\mathcal{S}$ is not already in $\mathbf{Q}$
        $\mathbf{Q} \leftarrow \mathbf{Q} \cup \mathcal{S}$
      end if
      for $j \in \{1, \dots, M\}$ **do**
        if USLP($\mathcal{S}, j$) **then** ▷ Check neighbors
          $\mathcal{S}' \leftarrow \mathcal{S} \triangle \{j\}$
          $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{S}'$ ▷ Add to the queue
        end if
      end for
    end if
  end while
  Return Q
end procedure

Proof. Suppose there exists $\mathbf{x}' \in \mathcal{D} \setminus \left(\bigcup_{\mathcal{S} \in \mathbf{Q}} \mathcal{X}(\{\mathbf{S}\})\right)$. Since $\mathcal{D}$ is connected, there exists a path γ such that $\mathbf{x}' \in \gamma$. Let $\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_K$ denote a sequence of activated sets where $\gamma \subseteq \bigcup_{i=0}^{K} \mathcal{X}(\{\mathcal{S}_i\})$. Given the finite neurons in the ReLU SNCBF, the breadth-first search in the activated set is complete [133]. By the completeness, we have for any $\mathbf{T}' \subseteq \mathbf{T}(\mathcal{S}_0, \dots, \mathcal{S}_K)$, $\left(\bigcap_{l=0}^{K} \mathcal{X}(\{\mathcal{S}_l\})\right) \cup \mathbf{T}' \subseteq \{\mathcal{X}(\{\mathcal{S}_1\}), \dots, \mathcal{X}(\{\mathcal{S}_r\})\}$. Since $\mathbf{x}' \in \mathcal{D} \setminus \left(\bigcup_{\mathcal{S} \in \mathbf{Q}} \mathcal{X}(\{\mathbf{S}\})\right)$, we have $\mathbf{x}' = \gamma(t')$ and $\mathbf{x}^* = \gamma(t)$ for some $t \in [0, t']$ such that $\mathbf{x}^* \in \mathbf{T}'$. Then, we have the hinge $\mathbf{T}' \nsubseteq \mathbf{T}(\mathcal{S}_0, \dots, \mathcal{S}_K)$, and $\mathbf{T}' \nsubseteq \{\mathcal{X}(\{\mathcal{S}_1\}), \dots, \mathcal{X}(\{\mathcal{S}_r\})\}$, thus creating a contradiction. □

With all activated sets included in $\mathbf{Q}$, we next verify each $\mathcal{S} \in \mathbf{Q}$.

Correctness Verification

We first verify the correctness condition in Definition 5.5.3. We denote the state $\mathbf{x} \in \mathcal{D} \subseteq \mathcal{X}_s$ with $h(x) < 0$ as a correctness counterexample. Let $\mathbf{Q} : \{\mathcal{S} : \mathcal{X}(\{\mathcal{S}\}) \cap \mathcal{D} \neq \emptyset\}$. We need to show that there does not exist any correctness counterexamples for all hyperplanes $\mathcal{X}(\{\mathcal{S}\})$ for all $\mathcal{S} \in \mathbf{Q}$ such that $\mathcal{X}_U \cap \mathcal{X}(\{\mathcal{S}\}) = \emptyset$. It suffices to solve

the nonlinear program

$$\begin{array}{ll} \min_{\mathbf{x}} & h(\mathbf{x}) \\ \text{s.t.} & \overline{W}_{1j}(\{\mathcal{S}\})^T\mathbf{x} + \overline{r}_{1j}(\{\mathcal{S}\}) \geq 0 \ \forall j \in \mathcal{S} \\ & \overline{W}_{1j}(\{\mathcal{S}\})^T\mathbf{x} + \overline{r}_{1j}(\{\mathcal{S}\}) \leq 0 \ \forall j \notin \mathcal{S} \\ & \overline{W}(\{\mathcal{S}\})^T\mathbf{x} + \overline{r}(\{\mathcal{S}\}) \geq 0. \end{array} \tag{5.40}$$

If the optimal value is negative, we return the verification result $False$ and we collect the solution $\mathbf{x}^*$ as a counterexample, denoted as $\hat{\mathbf{x}}_{ce}$. Otherwise, we return the verification result $True$ and an empty counterexample placeholder, i.e., $Null$.

Feasibility Verification

Next, we verify the feasibility condition in Definition 5.5.3. The function $\tilde{B}$ is a continuous piece-wise linear function characterized by the activation set $\mathcal{S}(t)$. Let $\tilde{z}(z_{1js}) := \mathbb{1}(z_{1js} > 0) - \frac{1}{2}\text{sgn}(z_{1js})$ We define $\lambda^{\mathcal{S}}(\mathbf{x}_t)$ and $\xi^{\mathcal{S}}(\mathbf{x}_t)$ as follows, when $\mathcal{S}(t) = \mathcal{S}$.

$$\begin{aligned} \lambda^{\mathcal{S}}(\mathbf{x}_t) :=& \frac{1}{2}\sum_{j=1}^{M}\left(W_{2j}\tilde{z}(z_{1js})W_{1j}^T - \frac{|W_{2j}|}{R_j}\zeta(\mathbf{x}_t)\right)g(\mathbf{x}_t) \\ \xi^{\mathcal{S}}(\mathbf{x}_t) :=& \frac{1}{2}\sum_{j=1}^{M}\left(W_{2j}\tilde{z}(z_{1js})W_{1j}^T - \frac{|W_{2j}|}{R_j}\zeta(\mathbf{x}_t)\right)f(\mathbf{x}_t) \\ & - \frac{1}{2}\frac{|W_{2j}|}{R_j}\mathbf{tr}(\sigma_s^T W_{1j}W_{1j}^T\sigma_s). \end{aligned}$$

We consider the case with control input constraints $u \in \mathcal{U} := \{u : Au \leq \mathbf{b}\}$. Let $\mathbf{Q} : \{\mathcal{S} : \mathcal{X}(\{\mathcal{S}\}) \cap \mathcal{D} \neq \emptyset\}$. The following proposition presents the feasibility condition for a ReLU SNCBF.

Proposition 5.5.5. *Let $\Lambda^{\mathcal{S}}(\mathbf{x}_t)$ and $\Xi^{\mathcal{S}}(\mathbf{x}_t)$ be as follows.*

$$\Lambda^{\mathcal{S}}(\mathbf{x}_t) := \begin{bmatrix} -\lambda^{\mathcal{S}}(\mathbf{x}_t) \\ A \end{bmatrix} \qquad \Xi^{\mathcal{S}}(\mathbf{x}_t) := \begin{bmatrix} \xi^{\mathcal{S}}(\mathbf{x}_t) \\ \mathbf{b} \end{bmatrix}.$$

For all $\mathbf{x} \in \tilde{\mathcal{D}}$, there always exists a u satisfying (5.21) if and only if, for all

$\mathcal{S} \in \mathbf{Q}$, *there is no* $\mathbf{x} \in \tilde{\mathcal{D}} \cap \mathcal{X}(\{\mathcal{S}\})$ *and* $\mathbf{y} \in \mathbb{R}^{n_u+1}$ *such that*

$$
\begin{aligned}
[\mathbf{y}]_1 \geq & 0, \ \forall i \in \{1, \ldots, n_u + 1\} && \text{(5.41a)} \\
-\mathbf{y}^T \Xi^{\mathcal{S}}(\mathbf{x}) < & 0 && \text{(5.41b)} \\
\left[\mathbf{y}^T \Lambda^{\mathcal{S}}(\mathbf{x})\right]_1 = & 0, \ \forall i \in \{1, \ldots, n_u + 1\}. && \text{(5.41c)}
\end{aligned}
$$

The proof is similar to Proposition 5.4.5.

In a similar fashion to the correctness conditions, the feasibility condition (5.41) can be verified by solving the nonlinear program

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & \mathbf{y}^T \begin{pmatrix} \xi^{\mathcal{S}}(\mathbf{x}) \\ \mathbf{b} \end{pmatrix} \\
\text{s.t.} \quad & \mathbf{y}^T \begin{pmatrix} -\lambda^{\mathcal{S}}(\mathbf{x}) \\ A \end{pmatrix} = \mathbf{0} \\
& \mathbf{y} \geq \mathbf{0} \\
& \tilde{B}(\mathbf{x}) \geq 0 \\
& \overline{W}_{1j}(\{\mathcal{S}\})^T \mathbf{x} + \overline{r}_{1j}(\{\mathcal{S}\}) \geq 0 \ \forall j \in \mathcal{S} \\
& \overline{W}_{1j}(\{\mathcal{S}\})^T \mathbf{x} + \overline{r}_{1j}(\{\mathcal{S}\}) < 0 \ \forall j \notin \mathcal{S}
\end{aligned}
\tag{5.42}
$$

and checking whether the optimal value is nonnegative (safe) or negative (unsafe). If the optimal value is nonnegative, we have the verification result $True$. Otherwise, we have the verification result $False$ and we collect the solution $\mathbf{x}^*$ as a counterexample, denoted as $\hat{\mathbf{x}}_{ce}$.

To enhance efficiency, we present sufficient conditions for verification. Since the ReLU SNCBF is smooth in each hyperplane $\mathcal{X}(\mathcal{S})$, we use the sufficient conditions derived in Section 5.4.2. Under the condition of Corollary 5.4.6, the nonlinear program (5.42) reduces to

$$
\begin{aligned}
\min_{\mathbf{x}} \quad & \xi^{\mathcal{S}}(\mathbf{x}) \\
\text{s.t.} \quad & \tilde{B}(\mathbf{x}) \geq 0 \\
& \mathbf{x} \in \mathcal{X}(\{\mathcal{S}\}) \\
& \left[\lambda^{\mathcal{S}}(\mathbf{x})\right]_i = 0, \ \forall i \in \{1, \ldots, n_u\}.
\end{aligned}
\tag{5.43}
$$

The system with $u = \mathbf{0}$ is feasible in hyperplane $\mathcal{X}(\{\mathcal{S}\})$ if $\xi^{\mathcal{S}}(\mathbf{x}) \geq 0$. It suffices to verify feasibility by solving the following nonlinear program.

$$
\begin{aligned}
\min_{\mathbf{x}} \quad & \xi^{\mathcal{S}}(\mathbf{x}) \\
\text{s.t.} \quad & \tilde{B}(\mathbf{x}) \geq 0 \\
& \mathbf{x} \in \mathcal{X}(\{\mathcal{S}\}).
\end{aligned}
\tag{5.44}
$$

The ReLU SNCBF verification is summarized in Algorithm. 6. Given an input $\mathbf{x}_0 \in \mathcal{D}$, the verification determines if the given ReLU SNCBF B is valid by checking if the conditions in Definition 5.5.3 are satisfied. If so, the verification returns result $True$, otherwise, it outputs $False$ and the counterexamples $\hat{\mathbf{x}}_{ce}$ where the verification fails. The verification checks if B suffices to satisfy (5.42) by sequentially examining the sufficient conditions (5.44) and (5.43) for better efficiency.

Algorithm 6 ReLU SNCBF Verification

Require: **Inputs:** Candidate ReLU SNCBF B, stochastic dynamics f, g, σ, safe and unsafe sets $\mathcal{X}_S, \mathcal{X}_U$, seed state $\mathbf{x}_0 \in \mathcal{D}$

Ensure: **Outputs:** Verification result res and counterexample $\hat{\mathbf{x}}_{ce}$ if verification fails

```
procedure VERIFICATION(S(x_0))
    S ← Enumeration(S_0)                          ▷ Algorithm. 5
    for S ∈ Q do
        res, x̂_ce ← Solve (5.40)                  ▷ Correctness Verification
        res, x̂_ce ← Solve (5.44)                  ▷ Feasibility Verification
        if ¬res then
            res, x̂_ce ← Solve (5.43)
            if ¬res then
                res, x̂_ce ← Solve (5.42)          ▷ Exact Condition
            end if
        end if
    end for
    Return res, x̂_ce
end procedure
```

Synthesis with Verification

The synthesis with verification of ReLU SNCBF shares the same loss function structure (5.18) as the smooth case. For each sample $\hat{\mathbf{x}} \in \mathcal{T}$ the safe control signal u is calculated using (5.39). The correctness loss $\mathcal{L}_c(\mathcal{T})$ penalizes the counterexample with negative minima of (5.40) (correctness condition of Definition 5.4.4) defined as

$$\mathcal{L}_c = \frac{1}{N} \sum_{\hat{\mathbf{x}}_i \in \mathcal{X}_s} \max\left(0, \left(\epsilon - \tilde{B}(\hat{\mathbf{x}}_i)\right) \mathbb{1}_{\mathcal{X}_s}\right) + \frac{1}{N} \sum_{\hat{\mathbf{x}}_j \in \mathcal{X}_U} \max\left(0, \left(\tilde{B}(\hat{\mathbf{x}}_j) + \epsilon\right) \mathbb{1}_{\mathcal{X}_U}\right). \quad (5.45)$$

The loss $\mathcal{L}_f$ enforces the satisfaction of the feasibility constraint by inserting a positive relaxation term r in the constraint and minimizing r with a large penalization in the objective function, penalizing the violations of constraint (5.41a)-(5.41c) (feasibility condition of Definition 5.5.3), defined as $\mathcal{L}_f = ||u(\hat{\mathbf{x}}) - u_{ref}(\hat{\mathbf{x}})||^2 + r$.

We note that the ReLU SNCBF approach presented in this section is designed for single hidden ReLU layer networks.

5.6 Experiments

In this section, we evaluate our proposed methods experimentally on 3 different systems. We first present the experiment settings, including hardware details and dynamical models for each experiment case. We evaluate the verifiable synthesis of a smooth SNCBF on the inverted pendulum and the unicycle model. We validate VITL synthesis of ReLU SNCBF on the Darboux model and the unicycle model. Finally, we compare smooth and ReLU SNCBF on the unicycle model against a baseline method.

5.6.1 Experiment Settings

The learning experiments of the SNCBFs are conducted on a computing platform equipped with an AMD Ryzen Threadripper PRO 5955WX CPU, 128GB RAM, and two NVIDIA GeForce RTX 4090 GPUs. We implement the neural networks in PyTorch, train them with the SGD optimizer, and verify them using dReal. Our code is available at https://github.com/HongchaoZhang-HZ/StochasticNCBF. During SNCBF training, we sample states uniformly from $\mathcal{X}$ without deploying the system or collecting trajectories. The predefined state space $\mathcal{X}$ corresponds to the physical operating region of each system. During deployment, the SNCBF-based controller ensures that trajectories initialized in $\mathcal{X}_I \subseteq \mathcal{X}_S$ remain within the safe set $\mathcal{X}_S$ with high probability. If the state exits $\mathcal{X}$, the safety guarantees no longer apply. Next, we present the dynamic models and detailed settings of three cases, namely, the inverted pendulum, the Darboux, and the unicycle model.

Inverted Pendulum

We consider a continuous time stochastic inverted pendulum dynamics given as follows:

$$d\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \left(\begin{bmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u \right) dt + \sigma \, dW_t, \tag{5.46}$$

where $\theta \in \mathbb{R}$ denotes the angle, $\dot{\theta} \in \mathbb{R}$ denotes the angular velocity, $u \in \mathbb{R}$ denotes the controller, m denotes the mass and l denotes the length of the pendulum. We let the mass $m = 1$kg, length $l = 10$m and the diffusion matrix $\sigma = \mathsf{diag}(0.1, 0.1)$. We emphasize that this diffusion term is not a mere simulation artifact: the Itô correction

$\frac{1}{2}\mathsf{tr}\left(\sigma^\intercal \frac{\partial^2 B}{\partial \mathbf{x}^2}\sigma\right)$ appears explicitly in both the safety condition (5.8) and the verification programs, so the magnitude of σ directly enters the learned barrier condition, the certified margin ψ, and the synthesized controller. The sensitivity study in Section 5.6 evaluates the resulting behaviour over three diffusion magnitudes. The inverted pendulum operates in a state space and is required to stay in a limited safe stable region. The state space, initial safe region and the unsafe region are given as follows.

$$\begin{aligned}
\mathcal{X} &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]^2\right\} \\
\mathcal{X}_I &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in \left[-\frac{\pi}{15}, \frac{\pi}{15}\right]^2\right\} \\
\mathcal{X}_S &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]^2\right\} \\
\mathcal{X}_U &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_S\right\}.
\end{aligned}$$

Darboux

We consider the Darboux system [194], a nonlinear open-loop polynomial system

The dynamic model of Darboux is given as follows.

$$d\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 + 2x_1x_2 \\ -x_1 + 2x_1^2 - x_2^2 \end{bmatrix} dt + \sigma\ dW_t. \tag{5.47}$$

We define state space, initial region, and safe region as follows.

$$\begin{aligned}
\mathcal{X} &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in [-2, 2]^2\right\} \\
\mathcal{X}_I &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in [0, 1] \times [1, 2]\right\} \\
\mathcal{X}_S &= \left\{\mathbf{x} \in \mathbb{R}^2 : x_1 + x_2^2 \geq 0\right\} \\
\mathcal{X}_U &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_S\right\}
\end{aligned}$$

Unicycle Model

We evaluate our proposed method on a controlled system [14]. We consider vehicles (UAVs) to avoid a pedestrian on the road. The system state consists of a 2-D position and aircraft yaw rate $\mathbf{x} := [x_1, x_2, \psi]^T$. The system is manipulated by the yaw rate input u.

$$d\begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix} = \left(\begin{bmatrix} v\cos\psi \\ v\sin\psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u\right) dt + \sigma\ dW_t, \tag{5.48}$$

where $[x_p, y_p, \Theta]^T \in \mathcal{X} \subseteq \mathbb{R}^3$ is the state consisting of the location (x_p, y_p) of the robot and its orientation Θ and u_1, u_2 controlling its speed and orientation.

We define the state space, initial region, safe region, and unsafe region as follows.

$$\begin{aligned}
\mathcal{X} &= \left\{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} \in [-2, 2]^3\right\} \\
\mathcal{X}_I &= \left\{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} \in [-0.1, 0.1] \times [-2, -1.8] \times \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]\right\} \\
\mathcal{X}_S &= \left\{\mathbf{x} \in \mathbb{R}^3 : \mathcal{X} \setminus \mathcal{X}_U\right\} \\
\mathcal{X}_U &= \left\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \in [-0.2, 0.2]^2 \times [-2, 2]\right\}.
\end{aligned}$$

5.6.2 Experiment Results

We now present the results of our SNCBF experiments for both the Smooth and ReLU cases. For each experiment, we present the hyper-parameters, settings, and experiment results. Finally, we compare our SNCBFs with a state-of-the-art approach as a baseline.

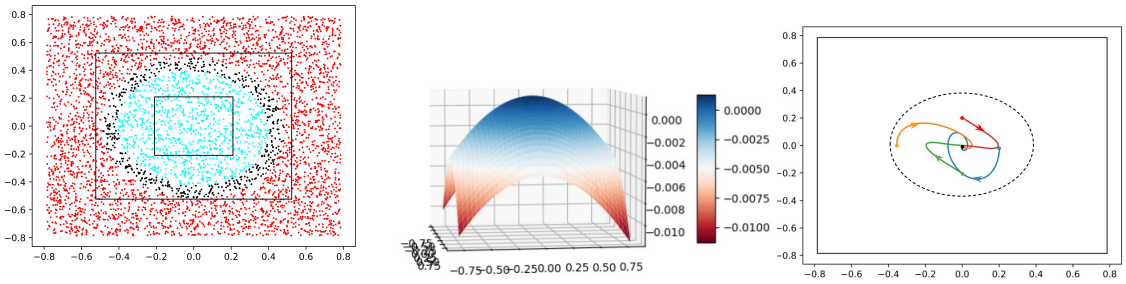
Smooth SNCBF Evaluation

For the class $\mathcal{K}$ function in the CBF inequality, we chose $\alpha(B) = \gamma B$, where $\gamma = 1$. We train the SNCBF assuming knowledge of the model. The SNCBF B consists of one hidden layer of 20 neurons, with Softplus activation function ($log(1 + exp(\mathbf{x}))$.

Inverted Pendulum: For the case of the inverted pendulum system, we set the training hyper-parameters to $\bar{\epsilon} = 0.00016$, $L_h = 0.01$, $L_{dh} = 0.4$ and $L_{d2h} = 2$ yielding $L_{\max} = 2.4$. We performed the training and simultaneously minimized the loss functions $\mathcal{L}_\theta, \mathcal{L}_M$, and $\mathcal{L}_v$. The training algorithm converged to obtain the SNCBF $B(\mathbf{x})$ with $\psi^* = -0.00042$. Thus, using Theorem 5.4.2, we can verify that the SNCBF obtained is valid, which ensures safety.

Visualizations of the trained SNCBF are presented in both 2D with sample points (Fig. 5.2a) and in 3D function value heat map (Fig. 5.2b). These visualizations demonstrate the successful separation of the initial safety region boundary from the unsafe region boundary. We validate our SNCBF-QP based safe controller on an inverted pendulum model with PyBullet. Fig 5.2c shows trajectories initiated inside the safe set (with different reference controllers) never leaving the safe set, thus validating our approach.

Unicycle Model: We then validate our smooth SNCBF-based safe control in the CARLA simulation environment. The vehicle is modeled as the unicycle model



(a) (b) (c)

Figure 5.2: This figure presents the experimental results on the inverted pendulum system. Fig. 5.2a visualizes of $B(\mathbf{x})$ over $\mathcal{X}$. Blue and red regions denote the safe region ($B \geq -\psi^*$) and the unsafe region ($B < \psi^*$), respectively. The initial safety region boundary and unsafe region boundary is denoted by black boxes. We observe that the boundary of trained SNCBF (black dots) successfully separate the unsafe and safe region. Fig. 5.2b shows the 3D plot of $B(\mathbf{x})$ over $\mathcal{X}$. Fig. 5.2c presents trajectories initiating inside the safe set following SNCBF-QP, following different reference controllers

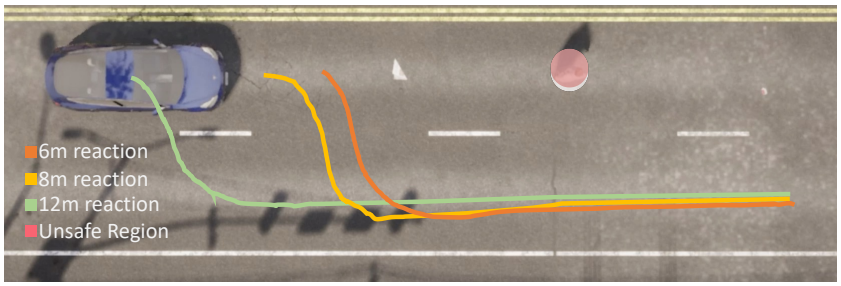


Figure 5.3: Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed SNCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

and its controller steers to avoid the pedestrian. If the control policy guides the vehicle to another lane in which there are no incoming obstacles, it then switches to an autonomous driving algorithm. The training hyper-parameters are set to $\bar{\epsilon} = 0.01$, $L_h = 1$, $L_{dh} = 1$ and $L_{d2h} = 2$ resulting in $L_{\max} = 4$. We performed the training which simultaneously minimized the loss functions $\mathcal{L}_\theta, \mathcal{L}_M$, and $\mathcal{L}_v$. The training algorithm converged to obtain the SNCBF $B(\mathbf{x})$ with $\psi^* = -0.04002$.

The trajectory of the vehicle is shown in Fig. 5.3. We show the trajectories of the proposed SNCBF-based safe control in different reaction distances, namely, 6, 8, and 12 meters, respectively. All three trajectories of the vehicle under control show that our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

Table 5.1: Sensitivity of the smooth unicycle SNCBF to the diffusion magnitude. The $\sigma = 0$ entry reproduces the deterministic Chapter 3 result.

Diffusion matrix σ	ψ^*	Certified coverage	Safety rate
0	-0.04002	77.85%	100%
$0.1I$	-0.04002	77.6%	100%
$0.2I$	-0.04000	77.8%	100%

Diffusion sensitivity. We repeat the smooth SNCBF unicycle experiment for isotropic diffusion matrices $\sigma \in \{0, 0.1I, 0.2I\}$ while retaining the same model, architecture, state domain, and training configuration. Table 5.1 summarizes the resulting margin, certified safe-set coverage, and empirical safety rate.

Each row corresponds to a separate certificate trained and evaluated at its own diffusion level; the experiment does not apply different noise levels post hoc to a single fixed certificate. Across the tested values, the certified margin and safe-set coverage remain nearly unchanged, and the controller achieves a 100% safety rate in all three cases. The near-flat values of ψ^* arise because this margin is dominated by the Lipschitz-cover slack $L_{\max}\bar{\epsilon}$, rather than serving as a sensitive readout of the diffusion magnitude. Thus, for this unicycle configuration, the smooth SNCBF synthesis is empirically stable over the considered diffusion range, while the conservatism of the certificate is primarily set by the cover-based bound.

ReLU SNCBF Evaluation

We next evaluate the verification-in-the-loop synthesis of ReLU SNCBF.

Darboux: We set $\lambda_f = 4.0$, $\lambda_c = 1.0$ and learning rate $l_r = 10^{-4}$. In the Darboux case, the neural network architecture utilized was a three-layer network consisting of 16 neurons in the hidden layer with ReLU activation functions, followed by a single output neuron. The training passed the verification at 86 epochs, achieving coverage of 54.4% of the safe region. The verification process was completed in 0.6 seconds. The training of the ReLU SNCBF required approximately 5.0 minutes.

Unicycle Model: We set $\lambda_f = 2.0$, $\lambda_c = 1.0$ and learning rate $l_r = 0.5 \times 10^{-4}$. In the unicycle model case, the complexity of the neural network architecture increased to a three-layer configuration, incorporating 16 hidden neurons with ReLU, again followed by a single output neuron. Training this network required 95 epochs to pass the verificaiton. Verification for the unicycle model took slightly longer, completing in 3.1 seconds with a safe region coverage of 41.2%. The corresponding synthesis time was approximately 10.5 minutes.

Case	Darboux	unicycle model
NN Architecture	2-16-φ-1	3-16-φ-1
Num Epoch	86	95
Safe Region Coverage	54.4%	41.2%
Verification Time	0.6 seconds	3.1 seconds
Synthesis Time	5.0 minutes	10.5 minutes

Table 5.2: ReLU SNCBF Synthesis Comparison

Table 5.2 summarizes the performance of the proposed ReLU SNCBF approach in verification and synthesis tasks across two distinct scenarios: the Darboux and the unicycle model. The verification time increases with the number of hyperplanes, which is jointly determined by the network size and system dimension.

SNCBF Comparison

	Baseline FT-SNCBF	Verifiable Smooth SNCBF	ReLU SNCBF
Verifiability	No	Yes	Yes
Coverage	14.8%	72.5%	45.7%
Training Time	39.4 minutes	64.2 minutes	17.6 minutes

Table 5.3: Comparison of Baseline, Smooth and ReLU SNCBF Synthesis. The smooth SNCBF is the proposed verifiable synthesis in Algorithm 4. The ReLU SNCBF is synthesized by VITL with the efficient verifier proposed in Algorithm 6.

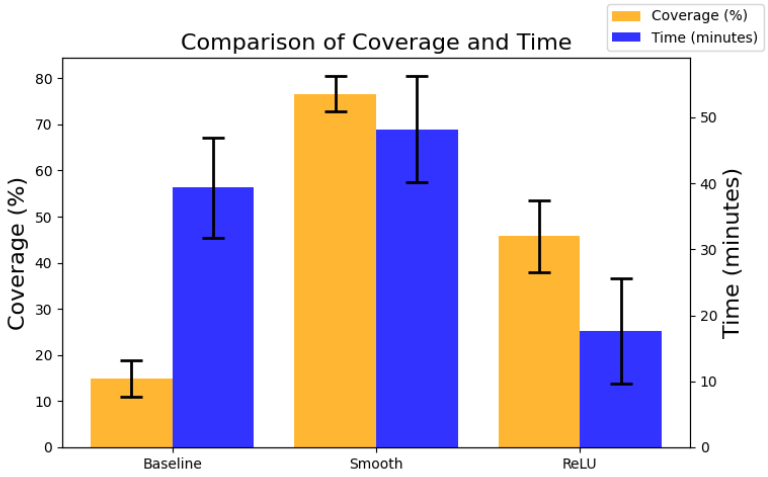


Figure 5.4: Comparison of coverage and training time across Baseline, Smooth and ReLU SNCBF Synthesis. Error bars indicate standard deviations across 5 seeds.

We compare our proposed method with an SNCBF trained with the baseline proposed in [195]. We proceed with 5 different random seeds for each synthesis method.

We compare in three aspects, verifiably safety, coverage of the safe region and the training time. The comparison results are summarized in Table 5.3 and Figure 5.4.

We observe that all synthesized SNCBFs successfully separate the safe and the unsafe regions.The baseline Fault-Tolerent SNCBF (FT-SNCBF) method did not guarantee verifiable results and achieved only 14.8% coverage after training for 39.4 minutes. The proposed verifiable safe synthesis for smooth SNCBF achieves superior safe region coverage (72.5%) with longer training time (64.2 minutes). The proposed ReLU SNCBF approach achieves a coverage of 45.7% and reduces training time to 17.6 minutes due to efficient verification.

Sensitivity Analysis

To assess the impact of sampling density on SNCBF synthesis, we conducted an experiment on the unicycle benchmark varying the sample count N and measuring: (i) the optimal barrier-certificate bound ψ^*, (ii) synthesis time T_s, and (iii) safe-region coverage r_c. Table 5.4 presents the results. Recall that the validity condition (Theorem 5.4.2) requires $L_{\max}\bar{\epsilon} + \psi^* \leq 0$, where $\bar{\epsilon}$ is the radius of balls covering the domain. The results reveal a smooth trade-off: increasing N (decreasing $\bar{\epsilon}$) yields

N	5.4×10^7	2×10^6	2.5×10^5	3.12×10^4	2×10^3
ψ	-0.04	-0.13	-0.25	-0.49	-1.20
T_s	64.2	26.5	17.3	12.1	7.25
r_c	78.65%	78.20%	76.53%	76.72%	72.66%

Table 5.4: Sensitivity analysis of sampling budget N (and implicitly $\bar{\epsilon}$) on the unicycle system.

tighter certificates (smaller $|\psi^*|$) and marginally improved coverage, but increases computation time. In particular, coverage remains stable between 72% and 79% across more than four orders of magnitude variation in N, demonstrating robustness to sampling density. A moderate sampling budget ($N \approx 10^5$) provides a favorable balance between synthesis time and coverage.

We note that the ReLU SNCBF approach presented in this chapter is designed for single hidden ReLU layer networks.

5.7 Chapter Summary

This chapter extended the deterministic NCBF framework of Chapter 3 to stochastic systems and advanced it in three significant directions. First, we developed NLP-

based correctness and feasibility verification procedures that replace dense-sampling Lipschitz bounds with targeted counterexample identification. Second, we integrated these verifiers into a Verification-In-The-Loop (VITL) training loop that iteratively refines the barrier through counterexample-guided synthesis, enabling deeper network architectures and substantially larger certified safe sets. Third, we extended stochastic barrier certificates to ReLU-activated networks via Tanaka's formula, demonstrating that piecewise-linear barriers can provide meaningful probabilistic safety guarantees with exact enumeration-based verification.

Together with Chapters 3 and 4, this chapter completes Part I's treatment of learning verified neural safety certificates. Three complementary verification paradigms have been developed: Lipschitz-based training-time certification (Chapter 3), distribution-free conformal prediction (Chapter 4), and counterexample-guided VITL synthesis (this chapter). Each offers distinct trade-offs between conservatism, scalability, and the type of guarantee provided.

The diffusion sweep further shows that, for the cover-based smooth SNCBF synthesis, the certificate is limited primarily by the Lipschitz slack $L_{\max}\bar{\epsilon}$ rather than by sensitivity of ψ^* to the diffusion physics. This is the bottleneck removed by conformal calibration in Chapter 4, which replaces the global Lipschitz-cover correction with a distribution-relative finite-sample margin.

A natural question is whether the conformal machinery of Chapter 4 could replace worst-case verification here as well. It plausibly could provide probabilistic bounds on the violation rate of the stochastic CBF condition at far lower computational cost; however, exchangeability of calibration samples is delicate in closed loop—visited states depend on the policy being certified, inducing distribution shift—and the resulting guarantee would be weaker than the region-wise conditions verified here. We therefore view the two paradigms as complementary rather than competing: VITL supplies deterministic, region-exact checks wherever enumeration is tractable, while conformal calibration remains attractive for very high-dimensional systems where enumeration becomes prohibitive. A hybrid scheme that conformally calibrates the residual error left by incomplete region enumeration is a promising direction for future work.

Bridge to Part II. All three chapters in Part I enforce safety in a *single-step, myopic* sense: the barrier condition constrains the instantaneous rate of change of the certificate without anticipating future evolution. While this is sufficient for many applications, it can be overly conservative or infeasible when control authority is limited or future hazards are foreseeable.

Part II broadens the safety horizon. Chapter 6 introduces Collision Cone CBFs, which embed geometric planning primitives into barrier design to provide velocity-level collision avoidance with temporal look-ahead. Chapter 7 then replaces hand-designed barrier functions with learned ones, using offline reinforcement-learning value functions as surrogate reachability approximations to produce barrier certificates that account for long-horizon constraint satisfaction.

The results presented in this chapter are based on [197].

Part II

Towards Planning-Aware Safety

Standard CBF conditions enforce safety in a single-step, myopic sense. This part goes further: first through geometric planning primitives that provide velocity-level look-ahead for collision avoidance, then through value-function-guided barrier synthesis for general, long-horizon safe control.

Chapter 6

Collision Cone CBFs: Planning-Aware Safety for Robotics

6.1 Related Works

Control Barrier Functions (CBFs) have emerged as a powerful framework for enforcing safety in robotic systems by formulating safety as a forward-invariance condition on a designated safe set [8, 6]. Given a CBF, the safety-critical controller can be synthesized by solving a Quadratic Program (QP) at each time step, enabling real-time operation. This paradigm has been successfully applied to adaptive cruise control [7], bipedal walking [70], and multi-robot coordination [179].

A central challenge in applying CBFs to obstacle avoidance is the construction of appropriate barrier functions that encode collision-free conditions. Most existing approaches define CBFs based on Euclidean distance to obstacles [6, 179], which leads to position-level constraints that do not account for the relative velocity between the robot and the obstacle. For static obstacles, distance-based CBFs are effective; however, for *moving* obstacles, these formulations can be overly conservative, forcing the robot to maintain large clearances regardless of the closing velocity, or insufficiently safe when the obstacle approaches at high speed.

Higher-Order CBFs (HOCBFs) [186, 187] extend the basic CBF framework to systems with higher relative degree, enabling safety enforcement for acceleration-controlled or jerk-controlled platforms. Exponential CBFs [112] offer an alternative approach for handling relative degree. However, both families still rely on position-based safety functions that are agnostic to the velocity of the obstacle.

Several works have explored velocity-aware safety for obstacle avoidance. The collision cone, originally developed in the aerospace community for aircraft conflict resolution [26], encodes the geometric condition under which two objects are on a collision course by examining whether the relative velocity vector lies within a cone defined by the obstacle boundary. This concept has been used in reactive planning [54] but has not been systematically integrated into the CBF framework prior to this work.

6.2 Contributions of this Chapter

This chapter makes the following contributions:

1. We introduce *Collision Cone Control Barrier Functions (C3BFs)*, a new class of CBFs that encode velocity-level collision avoidance constraints by embedding the collision cone condition into the barrier function framework. Unlike distance-based CBFs, C3BFs explicitly reason about the relative velocity between the robot and moving obstacles, yielding less conservative and more effective safety filters.

2. We develop C3BF formulations for three canonical robotic platforms: the acceleration-controlled *unicycle*, the *bicycle* model, and the *quadrotor* model, covering both ground and aerial vehicles. For each platform, we derive the barrier function, verify the relative-degree conditions, and construct the resulting QP-based safety filter.

3. We provide extensive simulation and hardware experiments demonstrating collision avoidance with static and dynamic obstacles, including head-on, crossing, and multi-obstacle scenarios. We validate the approach on differential-drive UGVs and Crazyflie quadrotors, demonstrating real-time feasibility.

4. We extend the C3BF framework to *cluttered environments* via Polygonal Cone Control Barrier Functions (PolyC2BFs), which handle polygonal obstacle shapes and enable safe navigation in complex settings.

The results presented in this chapter are based on [170, 163, 154, 155, 153].

6.3 Preliminaries

In this section, we provide the relevant background necessary to formulate our problem of moving obstacle avoidance. Specifically, we first describe the vehicle models

considered in our work. Next, we formally introduce Control Barrier Functions and their importance for real-time safety-critical control for these types of vehicle models. Finally, we explain the shortcomings of existing CBF approaches in the context of collision avoidance of moving obstacles.

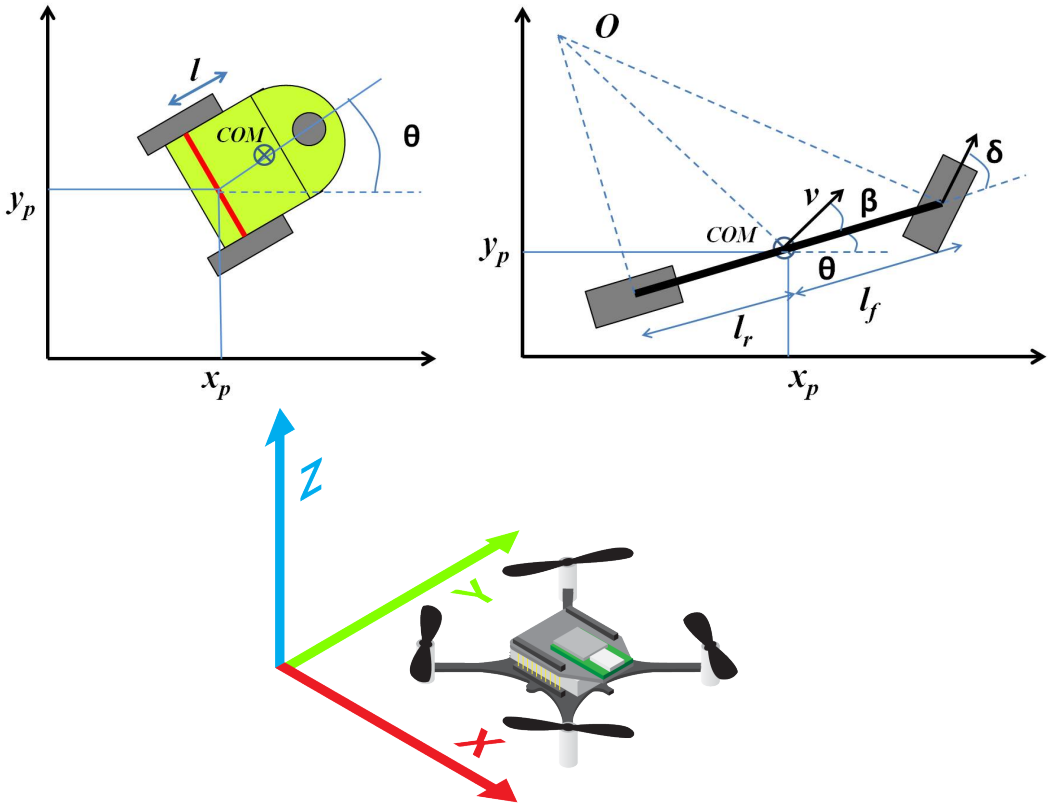


Figure 6.1: Schematic of Unicycle (left); Bicycle model (right); Quadrotor model (down).

6.3.1 Vehicle models

We specifically focus on three different models, namely the acceleration-controlled unicycle, bicycle, and quadrotor models, as explained next.

Unicycle Model

A unicycle model has state variables x_p, y_p, θ, v, ω that denote pose, linear velocity, and angular velocity, respectively. The control inputs are linear acceleration (a) and angular acceleration (α). In Fig. 6.1, we show a differential drive robot, which is modeled as a unicycle. The resulting dynamics of this model is shown below:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix}. \tag{6.1}$$

Although the commonly used unicycle model in the literature includes linear and angular velocities v, ω as input, we use accelerations as input. This is due to the fact that differential drive robots have torques as inputs to the wheels that directly affect accelerations. In other words, we can treat the force/acceleration applied from the wheels as input. As a result, v, ω become state variables in our model.

Bicycle Model

The bicycle model has two wheels, where the front wheel is used for steering (Fig. 6.1). This model is typically used for self-driving cars, where we treat the front and rear wheel sets as a single virtual wheel (for each set) by considering the difference of steer in the right and left wheels to be negligible [67, 129, 120].

Similar to [67], we assume that the slip angle is constrained to be small due to lateral acceleration. As a result, we approximate $\cos\beta \approx 1$ and $\sin\beta \approx \beta$. Consequently, we get the following affine dynamics of the bicycle model:

$$\underbrace{\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} v\cos\theta \\ v\sin\theta \\ 0 \\ 0 \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} 0 & -v\sin\theta \\ 0 & v\cos\theta \\ 0 & \frac{v}{l_r} \\ 1 & 0 \end{bmatrix}}_{g(x)} \underbrace{\begin{bmatrix} a \\ \beta \end{bmatrix}}_{u}. \tag{6.2}$$

x_p and y_p denote the coordinates of the vehicle's centre of mass (CoM) in an inertial frame. θ represents the orientation of the vehicle with respect to the positive x-axis. a is the linear acceleration at CoM. l_f and l_r are the distances of the front and rear axles from the CoM, respectively. δ is the steering angle of the vehicle and β is the vehicle's slip angle, i.e., the steering angle of the vehicle mapped to its CoM (Fig. 6.1). This should not be confused with the tyre slip angle.

Quadrotor Model

The quadrotor model has four propellers, which provides upward thrusts of (f_1, f_2, f_3, f_4) (Fig. 6.1) and the states needed to describe the quadrotor system is given by $x =$

$[x_p, y_p, z_p, \dot{x}_p, \dot{y}_p, \dot{z}_p, \Phi, \theta, \Psi, \omega_1, \omega_2, \omega_3]$. The quadrotor dynamics is as follows[57, 126]:

$$\underbrace{\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \\ \ddot{x}_p \\ \ddot{y}_p \\ \ddot{z}_p \\ \dot{\Phi} \\ \dot{\theta} \\ \dot{\Psi} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{z}_p \\ 0 \\ 0 \\ -g \\ \mathbf{W}^{-1} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \\ -I^{-1}\vec{\omega} \times I\vec{\omega} \\ . \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \frac{1}{m}\mathbf{R} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ I^{-1}L \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ c_\tau & -c_\tau & c_\tau & -c_\tau \end{bmatrix} \end{bmatrix}}_{g(x)} \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}}_{u}, \tag{6.3}$$

where x_p, y_p and z_p denote the coordinates of the base centre of the quadrotor in an inertial frame. Φ, θ and Ψ represents the (roll, pitch & yaw) orientation of the quadrotor. (Fig. 6.1). $\mathbf{R}$ is the rotation matrix (from the body frame to the inertial frame), m is the mass of the quadrotor, $\mathbf{W}$ is the transformation matrix for angular velocities from the inertial frame to the body frame, I is the inertia matrix, and L is the diagonal length of the quadrotor. c_τ is the constant that determines the torque produced by each propeller.

6.3.2 High-Order Control Barrier Functions

In the previous sections we have seen how CBFs can be used to enforce safety-critical constraints for nonlinear (control affine) systems, there is one critical restriction: the safety-critical constraints have been so far assumed to be of relative-degree one, i.e., the first time-derivative of the CBF has to depend on the control input. A popular way to overcome this challenge is via High-Order Control Barrier Functions (HO-CBFs)[186, 187]. In this approach, a candidate CBF is mechanically extended to a new function that may be used to enforce safety. The success of this technique relies on the notion of relative degree.

Definition 6.3.1 ([186, 187])**.** A smooth function $\psi : \mathbb{R}^n \to \mathbb{R}$ is said to have relative degree $r \in \mathbb{N}$ for (2.1) at $x \in \mathbb{R}^n$ if:

1. $L_g L_f^i \psi(x) = 0, \quad \forall i \in \{0, \dots, r-2\}$,
2. $L_g L_f^{r-1} \psi(x) \neq 0$.

To define HO-CBFs, consider a state constraint set $S \subset \mathbb{R}^n$ defined as the super level set of a smooth function $\psi : \mathbb{R}^n \to \mathbb{R}$. Assuming that ψ has relative degree $r \geq 2$ on S, define:

$$\psi_{i+1}(x) := L_f \psi_i(x) + \kappa_i(\psi_i(x)), \quad \forall i \in \{0, \dots, r-2\}, \tag{6.4}$$

where $\kappa_i \in \mathcal{K}_e$ (*class* $\mathcal{K}$ functions) are smooth, with $\psi_0(x) := \psi(x)$. This collection of functions produces a collection of sets:

$$S_i := \{x \in \mathbb{R}^n : \psi_i(x) \geq 0\}, \quad \forall i \in \{0, \dots, r-1\}. \tag{6.5}$$

These sets are used to define a candidate safe set as:

$$\mathcal{S} := \bigcap_{i=0}^{r-1} S_i \subseteq S, \tag{6.6}$$

which is a subset of the original constraint set $S = S_0$. The controlled invariance of this safe set can then be ensured through the existence of a HO-CBF.

Definition 6.3.2 ([186, 187])**.** A smooth function $\psi : \mathbb{R}^n \to \mathbb{R}$ defining a constraint set $S \subset \mathbb{R}^n$ is said to be a HO-CBF for (2.1) on a set $\mathcal{S} \subset S \subset \mathcal{D}$, if there exists an extended *class* $\mathcal{K}$ function κ such that for all $x \in \mathcal{D}$:

$$\sup_{u \in \mathbb{R}^m} \left\{ L_f \psi_{r-1}(x) + L_g \psi_{r-1}(x) u + \kappa(\psi_{r-1}(x)) \right\} \geq 0. \tag{6.7}$$

Any locally Lipschitz controller satisfying the above condition renders the set $\mathcal{S}$ forward invariant. Since $\mathcal{S} \subset S$, this ensures that trajectories remain within the constraint set S so long as they are defined.

Even though ψ satisfies the criteria to be considered a HO-CBF, it does not align with the conventional definition of a CBF. Specifically, the safe set is not derived from the superlevel set of ψ or ψ_{r-1}; instead, it is defined as the intersection of sets outlined in (6.6). Although HO-CBFs offer a general framework for enforcing safety constraints and allow flexibility in the choice of functions κ_i, the robustness and stability properties commonly associated with first-order CBFs are not guaranteed for

arbitrary selections of these functions. As demonstrated in [152], such properties can be recovered by appropriately selecting κ_i. Conversely, poorly chosen κ_i may result in overly conservative behavior or unnecessarily tight constraints. We will discuss the comparison of Higher Order CBF with our proposed approach in Section 6.4.5.

Next, we introduce a CBF formulation that uses geometric intuition while addressing the limitations mentioned above.

6.4 Collision Cone Control Barrier Function

A collision cone, defined for a pair of objects, is a set which can be used to predict the collision between the two objects based on the direction of their relative velocity [26]. The collision cone of an object pair represents the directions, which if traversed by either objects, will result in a collision between the two. We will treat the obstacles as ellipses with the vehicle reduced to a point; therefore, throughout the rest of the paper, the term collision cone will refer to this case with the ego-vehicle's center being the point of reference.

Consider an ego-vehicle defined by the system (2.1) and a moving obstacle (pedestrian, another vehicle, etc.). This is shown pictorially in Fig. 6.2. We define the velocity and positions of the obstacle w.r.t. the ego vehicle. We *over-approximated* the obstacle to be an ellipse and draw two tangents from the vehicle's centre to a conservative circle (of radius $r := \max(c_1, c_2) + \frac{w}{2}$ where w is the width of the vehicle) encompassing the ellipse, taking into account the ego-vehicle's dimensions. For a collision to happen, the relative velocity of the obstacle must be pointing towards the vehicle. Hence, the relative velocity vector must not be pointing into the pink shaded region EHI in Fig. 6.2, which is a cone. Let $\mathcal{C}$ be this set of safe directions for this relative velocity vector. If there exists a function $B : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ satisfying *Definition: 2.3.1* on $\mathcal{C}$, then we know that a Lipschitz continuous control law obtained from the resulting QP (2.11) for the system ensures that the vehicle won't collide with the obstacle even if the reference u_{ref} tries to direct them towards a collision course. This novel approach of avoiding the pink cone region gives rise to **Collision Cone Control Barrier Functions (C3BFs)**. We propose the following CBF candidate:

$$B(x,t) :=< p_{rel}, v_{rel} > + \|p_{rel}\| \|v_{rel}\| \cos\phi, \tag{6.8}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product, p_{rel} is the relative position vector of the body center of the vehicle (X) and the center of the obstacle(X_o), v_{rel} is the relative velocity of the body center of the vehicle $(\dot{X})$ and the center of the obstacle$(\dot{X}_o)$, ϕ is the half angle of the cone, the expression of $\cos\phi$ is given by

$\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|p_{rel}\|}$ (see Fig. 6.2). The constraint simply ensures that the angle between p_{rel}, v_{rel} is less than $180° - \phi$.

Remark 6.4.1. We assume that the obstacle position and velocity are a function of time t. Therefore, p_{rel} $(= X_o(t) - X)$ and v_{rel} $(= \dot{X}_o(t) - \dot{X})$ and thereby the barrier function B is a function of time t.

Remark 6.4.2 (Velocity Constraints via Standard Barrier Frameworks). One may ask whether the velocity-level information captured by (6.8) could instead be encoded directly in a standard (first-order) barrier framework. A position-only barrier $h(x) = \|p_{rel}\| - r$ has relative degree two for all three acceleration-controlled models considered here, so encoding closing-velocity information requires lifting through HO-CBFs or exponential CBFs with auxiliary functions and class-$\mathcal{K}$ tunings κ_i (Section 6.3.2). The lifted constraint then acts on $\dot{h}$ but retains no explicit geometric notion of the direction of v_{rel}, tends to be conservative for fast-approaching obstacles, and its behavior depends on the tuning of κ_i [152]. The collision cone embedding (6.8), by contrast, is a single relative-degree-one quantity in which the relative velocity enters natively: it becomes active exactly when the pair is on a collision course and relaxes automatically as the geometry separates. An empirical comparison with HO-CBF constructions on the same platforms is provided in Section 6.4.5.

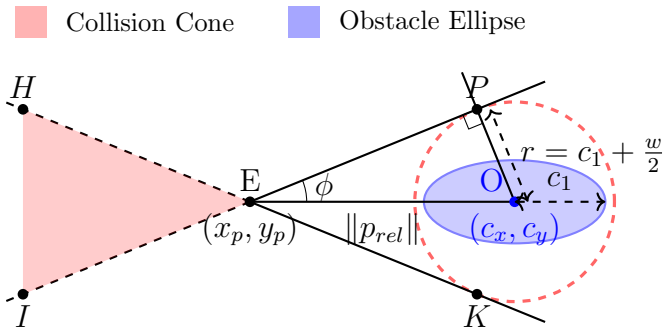


Figure 6.2: Construction of collision cone for an elliptical obstacle (where c_1 & c_2 are the major & minor axis length respectively) considering the ego-vehicle's dimensions (width: w).

6.4.1 Application to Unicycle

We first obtain the relative position vector between the body center of the unicycle and the center of the obstacle. Therefore, we have

$$p_{rel} := \begin{bmatrix} c_x(t) - (x_p + l\cos\theta) \\ c_y(t) - (y_p + l\sin\theta) \end{bmatrix} \tag{6.9}$$

Here l is the distance of the body center from the differential drive axis (see Fig. 6.1). We obtain its velocity as

$$v_{rel} := \begin{bmatrix} \dot{c}_x(t) - (v\cos\theta - \omega l \sin\theta) \\ \dot{c}_y(t) - (v\sin\theta + \omega l \cos\theta) \end{bmatrix}. \tag{6.10}$$

We have the following first result of the paper:

Theorem 6.4.3 ([170, 154]). *Given the acceleration controlled unicycle model (6.1) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = 0$), the proposed CBF candidate (6.8) with p_{rel}, v_{rel} defined by (6.9), (6.10) is a valid CBF defined for the set $\mathcal{D}$.*

Proof can be found in Appendix

6.4.2 Application to Bicycle

For the approximated bicycle model (6.2), we define the following:

$$p_{rel} := \begin{bmatrix} c_x(t) - x_p & c_y(t) - y_p \end{bmatrix}^T \tag{6.11}$$

$$v_{rel} := \begin{bmatrix} \dot{c}_x(t) - v\cos\theta & \dot{c}_y(t) - v\sin\theta \end{bmatrix}^T, \tag{6.12}$$

Here v_{rel} is NOT equal to the relative velocity $\dot{p}_{rel}$. However, for small β, we can assume that v_{rel} is the difference between obstacle velocity and the velocity component along the length of the vehicle $v\cos\beta \approx v$. In other words, the goal is to ensure that this approximated velocity v_{rel} is pointing away from the cone. This is an acceptable approximation as β is small and the obstacle radius chosen was conservative (see Fig. 6.2).

We have the following result:

Theorem 6.4.4 ([170, 154]). *Given the bicycle model (6.2) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = 0$), the proposed candidate CBF (6.8) with p_{rel}, v_{rel} defined by (6.11), (6.12) is a valid CBF defined for the set $\mathcal{C}$.*

Proof in the Appendix

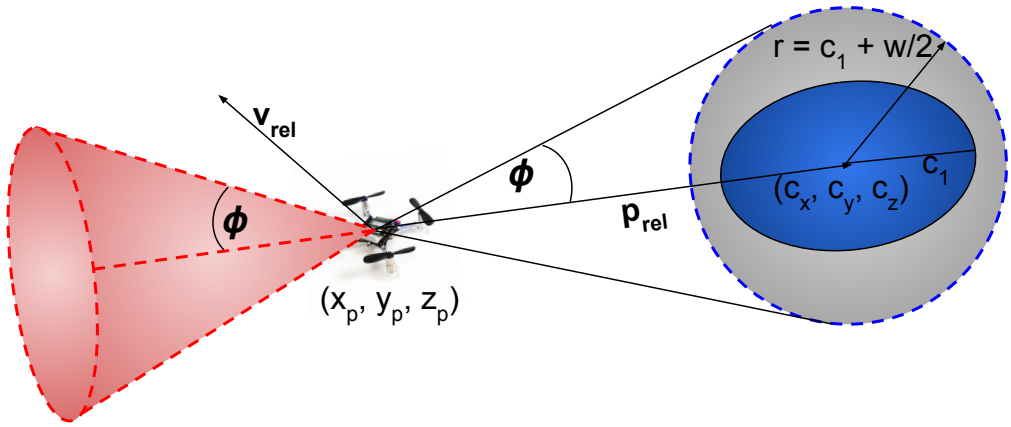


Figure 6.3: **3D CBF** candidate: The dimensions of the obstacle are comparable to each other, it can be assumed as a sphere

6.4.3 Application to Quadrotor

Based on the shape of the obstacle we can divide the proposed candidates into two cases: 1) When the dimensions of the obstacle are comparable to each other, we can assume the obstacle as a sphere with radius $r := \max(c_1, c_2, c_3) + \frac{w}{2}$, where w is the max width of the quadrotor absorbed in the obstacle width (shown in Fig. 6.3). We call the CBF candidate so formed in this case as **3D CBF** candidate (see Fig. 6.3). 2) When one of the dimensions is far bigger than the other dimensions, we can assume the obstacle as a cylinder with height $H := \max(c_1, c_2, c_3)$ and radius $r := \max_2(c_1, c_2, c_3) + \frac{w}{2}$ (where $\max_2$ is the second largest element in the list) and we call the candidate so formed in this case as **Projection CBF** candidate (see Fig. 6.4).

3D CBF candidate

When the dimensions of the obstacle are comparable to each other, we can assume the obstacle as a sphere with radius $r = \max(c_1, c_2, c_3) + \frac{w}{2}$, where w is the max width of the quadrotor absorbed in the obstacle width (shown in Fig. 6.3). We call the CBF candidate so formed in this case as **3D CBF** candidate (see Fig. 6.3). We first obtain the relative position vector between the body center of the quadrotor and the center of the obstacle. We have the following:

$$p_{rel} := \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} - \left(\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \right). \tag{6.13}$$

Here l is the distance of the body center from the base (see Fig. 6.1). c_x, c_y, c_z represents the obstacle location as a function of time. We obtain its relative velocity as

$$v_{rel} := \dot{p}_{rel}. \tag{6.14}$$

Now, we calculate the $\dot{v}_{rel}$ term which contains our inputs i.e. (f_1, f_2, f_3, f_4), as follows:

$$\dot{v}_{rel} = -\mathbf{R} \begin{bmatrix} 0 & \frac{Ll}{I_{yy}} & 0 & \frac{-Ll}{I_{yy}} \\ \frac{-Ll}{I_{xx}} & 0 & \frac{Ll}{I_{xx}} & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} + \text{additional terms.}$$

Please note that the additional terms in the above equation, refer to those terms that do not contain the input terms (f_1, f_2, f_3, f_4) and thus do not contribute to the calculation of $\mathcal{L}_g B$. We have the following result.

Theorem 6.4.5 ([163, 153]). *Given the quadrotor model (6.3) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = \ddot{c}_z = 0$), the proposed CBF candidate (6.8) with p_{rel}, v_{rel} defined by (6.13), (6.14) is a valid CBF defined for the set $\mathcal{D}$.*

Proof in the Appendix

Projection CBF candidate

When one of the dimensions is far bigger than the other dimensions, we can assume the obstacle as a cylinder with height $H := \max(c_1, c_2, c_3)$ and radius $r := \max_2(c_1, c_2, c_3) + \frac{w}{2}$ (where $\max_2$ is the second largest element in the list) and we call the candidate so formed in this case as **Projection CBF** candidate (see Fig. 6.4). Now, we have to consider the collision avoidance for elongated (cylindrical) obstacles. We have to obtain the relative position vector between the body center of the quadrotor and the intersection of the axis of the obstacle and the projection plane, where the projection plane is the plane perpendicular to the axis. Therefore, we have

$$(p_{rel})_{proj} := \mathcal{P}\left(\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} - \left(\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \right) \right). \tag{6.15}$$

Here l is the distance of the body center from the base (see Fig. 6.1). $\mathcal{P} : \mathbb{R}^3 \to \mathbb{R}^3$ is the projection operator, which can be assumed to be a constant[1]. Now, since the relative position lies on the projection plane, we have one more condition to satisfy:

$$< (p_{rel})_{proj}, \hat{n} >= 0, \tag{6.16}$$

where, $\hat{n}$ is the normal to the plane. Also, the relative velocity is given by:

$$(v_{rel})_{proj} := \frac{d(p_{rel})_{proj}}{dt} = (\dot{p}_{rel})_{proj}. \tag{6.17}$$

Now, we calculate the $\frac{d}{dt}(v_{rel})_{proj}$ term which contains our inputs i.e. (f_1, f_2, f_3, f_4), as follows:

$$\frac{d}{dt}(v_{rel})_{proj} = \mathcal{P}(-\mathbf{R} \begin{bmatrix} 0 & \frac{Ll}{I_{yy}} & 0 & \frac{-Ll}{I_{yy}} \\ \frac{-Ll}{I_{xx}} & 0 & \frac{Ll}{I_{xx}} & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} + \text{additional terms}).$$

or, from (6.15), we have

$$\frac{d}{dt}(v_{rel})_{proj} = \mathcal{P}(\dot{v}_{rel}), \tag{6.18}$$

where $\frac{d}{dt}(v_{rel})_{proj}$ is the projection of $\dot{v}_{rel}$ in (6.15) on the projection plane, that is:

$$(\dot{v}_{rel})_{proj} = \dot{v}_{rel} - < \dot{v}_{rel}, \hat{n} > \hat{n}. \tag{6.19}$$

Thus, from (6.16) and (6.19), we have the following:

$$< (p_{rel})_{proj}, (\dot{v}_{rel})_{proj} >=< (p_{rel})_{proj}, \dot{v}_{rel} > . \tag{6.20}$$

Similarly,

$$< (v_{rel})_{proj}, (\dot{v}_{rel})_{proj} >=< (v_{rel})_{proj}, \dot{v}_{rel} > . \tag{6.21}$$

We now provide the formal results for the Projection case of (6.8) in this subsection. The candidate is given as follows:

$$\begin{aligned} B_{proj}(x,t) =& < (p_{rel})_{proj}, (v_{rel})_{proj} > + \\ & \|(p_{rel})_{proj}\| \|(v_{rel})_{proj}\| \cos\phi, \end{aligned} \tag{6.22}$$

[1]Note that the obstacles are always translating and not rotating. In addition, it is not restrictive to assume that the translation direction is always perpendicular to the cylinder axis. This makes the projection operator a constant.

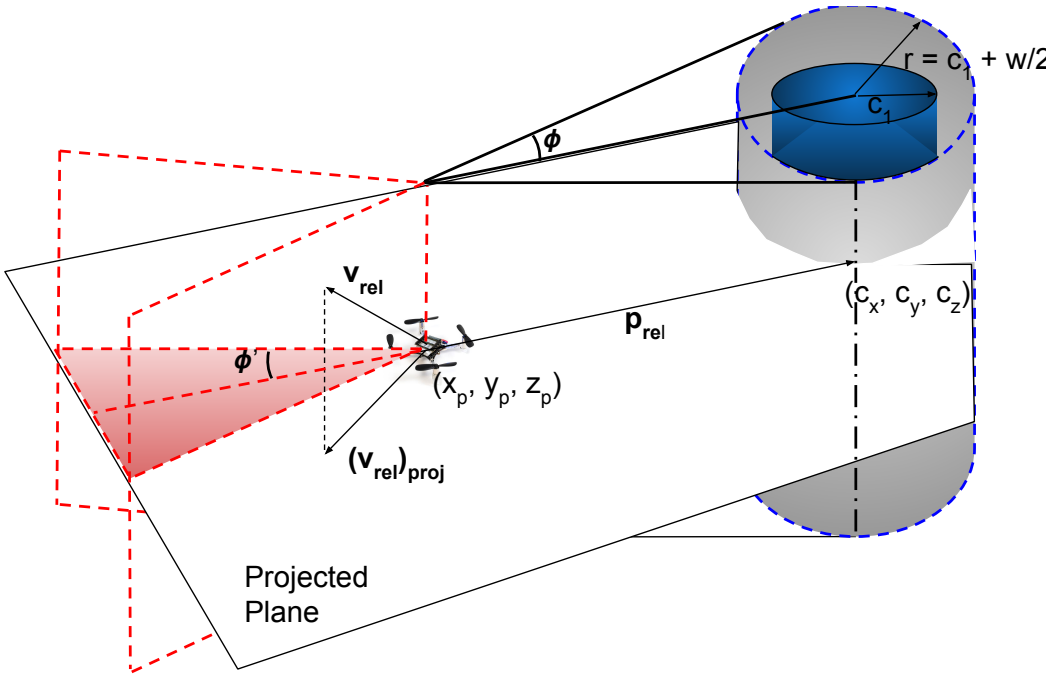


Figure 6.4: **Projection CBF** candidate: One of the dimensions, of the obstacle, is bigger than the other dimensions, it can be assumed as a cylinder.

where ϕ is the half angle of the cone, the expression of $\cos\phi$ is given by $\frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(p_{rel})_{proj}\|}$ (see Fig. 6.4). We now show that the proposed CBF candidate (6.22) is indeed a valid CBF.

Theorem 6.4.6 ([153]). *Given the quadrotor model (6.3) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = \ddot{c}_z = 0$), in a direction perpendicular to the cylindrical axis with no rotation, the proposed CBF candidate (6.22) with p_{rel}, v_{rel} defined by (6.15), (6.17) is a valid CBF defined for the set $\mathcal{D}$.*

Proof in the Appendix

6.4.4 Point mass model

Similar to Theorem 6.4.4, analogous results can be derived for simple point-mass models [93] of the form:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} u, \tag{6.23}$$

where $p = [x_p, y_p]^T$, $v = [v_x, v_y]^T$, and $u = [a_x, a_y]^T \in \mathbb{R}^2$ denotes the position, velocity and acceleration inputs, respectively. The point-mass extension follows by applying

Table 6.1: Comparison between the HOCBF (6.25) and the proposed C3BF (6.8) for different vehicle models.

CBFs	**Vehicle Models**	**Static Obstacle** (c_x, c_y, c_z)	**Moving Obstacle** $(c_x(t), c_y(t), c_z($
HOCBF	Unicycle (6.1)	Valid CBF, No steering	Valid CBF, but conservative
HOCBF	Bicycle (6.2)	Valid CBF	Not a valid CBF
HOCBF	Quadrotor (6.3)	Valid CBF	Valid CBF, but conservative
C3BF	Unicycle (6.1)	Valid CBF in $\mathcal{D}$	Valid CBF in $\mathcal{D}$
C3BF	Bicycle (6.2)	Valid CBF in $\mathcal{C}$	Valid CBF in $\mathcal{C}$
C3BF	Quadrotor (6.3)	Valid CBF in $\mathcal{D}$	Valid CBF in $\mathcal{D}$

† $(c_x(t), c_y(t), c_z(t))$ are continuous functions of time

the same CBF construction under the stated dynamics; a full proof is outside the scope of this chapter.

Remark 6.4.7. Theorems 6.4.3-6.4.6 offer safety guarantees in cases with a single obstacle. However, for multi-obstacle scenarios, the conditions outlined by these theorems might conflict.
To address this issue, we propose a more practical approach, incorporating a perception boundary that models the limits of the sensor's obstacle detection. The C3BF-QP activates when an obstacle enters this boundary, allowing the ego vehicle to prioritize nearby obstacles and reduce complexity in multi-obstacle environments. A detailed theoretical analysis of this approach is left for future work.

6.4.5 Comparison with Higher Order CBFs

In this section, we compare the Higher Order Control Barrier Functions (HO-CBFs) (introduced in section 6.3.2) with the proposed collision cone CBF. Now, let us consider a smooth function $\psi(x,t) := (c_x(t) - x_p)^2 + (c_y(t) - y_p)^2 + (c_z(t) - z_p)^2 - r^2$. Since, $L_g\psi(x,t) = 0$ and $L_gL_f\psi(x,t) \neq 0$, the relative degree of ψ is 2 ($r = 2$):

$$\begin{aligned} \psi_1(x,t) &:= L_f\psi(x,t) + \kappa_1(\psi(x,t)) \\ \psi_2(x,t) &:= L_f\psi_1(x,t) + L_g\psi_1(x,t)u + \kappa_2(\psi_1(x,t)), \end{aligned} \tag{6.24}$$

where κ_1, κ_2 are both extended class $\mathcal{K}$ functions. Let us examine the form of HO-CBF where κ_1 is a square root function (which is also strictly increasing), κ_2 is a linear function, due to its similarity to C3BF. Consequently, the resulting Higher Order CBF candidate takes the following form:

$$B_{HO}(x,t) := \psi_1(x,t) =< p_{rel}, v_{rel} > +\gamma\sqrt{(\|p_{rel}\|^2 - r^2)}. \tag{6.25}$$

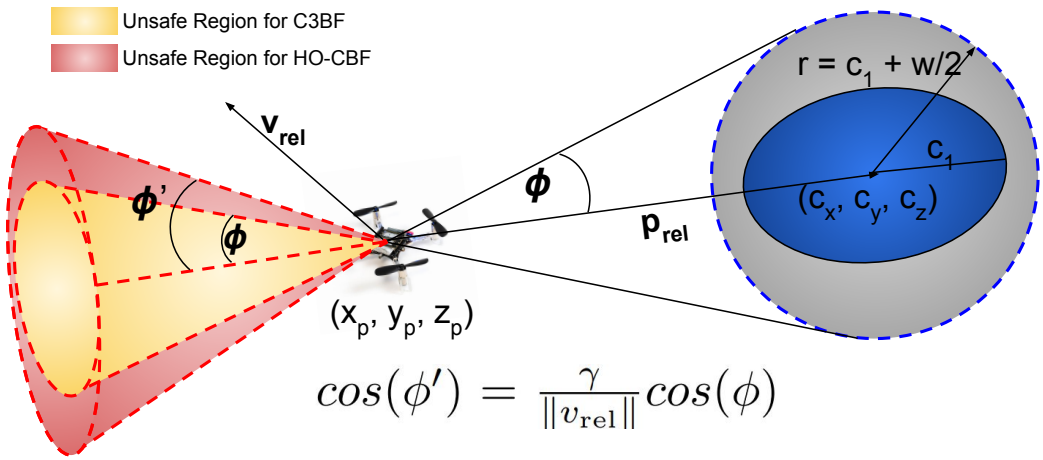


Figure 6.5: Comparison of HO-CBF with C3BF. Here we are trying to compare the ϕ' and ϕ obtained from the two CBF formulations. It can be observed that ϕ' (pink cone) is dependent on v_{rel}, while ϕ (yellow cone) is a constant, for a perticular distance from the obstacle.

with safe set defined as $\mathcal{S} := \{x \in \mathbb{R}^n : \psi(x) \geq 0\} \cap \{x \in \mathbb{R}^n : \psi_1(x) \geq 0\}$. Please note that the higher order CBF, is a general design technique that may require adjustments depending on the specific system dynamics and context. However, we consider the above HO-CBF candidate, due to its similarity with the proposed CBF.

We can show that the above-mentioned HO-CBF is a valid CBF for Unicycle and Quadrotor models (as we did in Theorems 6.4.3, 6.4.5 and 6.4.6), however, it is not a valid CBF for bicycle model. These results are summarised in Table-6.1. We will now compare it with the proposed collision cone CBF, for unicycle and agent models.

The C3BF concept aims to prevent the v_{rel} vector, which represents the relative velocity between the agent and the obstacle, from entering the collision cone region defined by the half-angle ϕ. Figure 6.5 illustrates this idea. We can rewrite the HO-CBF formula presented in (6.25) in the following form:

$$B_{HO}(x,t) =< p_{rel}, v_{rel} > + \|p_{rel}\|\|v_{rel}\| cos(\phi'), \tag{6.26}$$

where $cos(\phi') = \frac{\gamma}{\|v_{rel}\|} cos(\phi)$.

If we were able to identify a suitable γ (penalty term) for the given HO-CBF, it would result in a valid CBF as per [187]. When the relative velocity $\|v_{rel}\|$ is high, it leads to $\phi' > \phi$, causing an overestimation of the collision cone. Conversely, if $\|v_{rel}\|$ is low, it might result in $\phi' < \phi$, potentially violating the collision cone. In contrast, the C3BF approach allows the penalty term to vary over time, specifically by setting $\gamma = \|v_{rel}\|$, which enables a more accurate estimation of the collision cone compared to the HO-CBF method. Interestingly, this formulation can also be interpreted as a geometry-inspired instance of the HO-CBF framework, where the penalty functions

Table 6.2: Extreme unicycle comparison over 100 cases. Intervention energy is $E_{\text{int}} = \sum_k \|u_k - u_{\text{ref},k}\|_2^2 \Delta t$, which measures the cumulative squared control correction applied by the safety filter.

oprule Metric	C3BF	HO-CBF
Collisions	0/100	3/100
Mean intervention energy, E_{int}	0.347	2.441
Maximum $\lvert a \rvert$ [m/s^2]	3.590	17.895
Maximum $\lvert \alpha \rvert$ [rad/s^2]	10.363	15.076
Maximum $\|u\|_2$	10.363	22.459



Figure 6.6: Test setups: TurtleBot3 Burger, Stoch-Jeep, and Crazyflie 2.1, respectively.

κ_i are implicitly chosen based on the structure of the collision cone. This connection highlights that C3BF not only ensures formal safety guarantees but also uses domain-specific intuition to reduce conservativeness and improve practical applicability. This is also evident from the simulation outcomes of both CBFs, as demonstrated in Section 6.5.

To complement the geometric comparison, we evaluate C3BF and HO-CBF over the same 100 extreme unicycle cases. Table 6.2 reports collisions and the magnitude of the control intervention. The intervention energy is defined as the cumulative squared correction applied by the safety filter relative to the reference controller.

In these cases, C3BF avoided all collisions while HO-CBF recorded three collisions. C3BF also required less intervention energy and produced lower peak acceleration, angular acceleration, and control magnitude. These results support the geometric observation that matching the penalty to the relative velocity can reduce conservativeness and control intervention in the tested unicycle scenarios.

6.5 Results

In this section, we provide simulation and hardware experimental results to validate the proposed C3BF-QP. All simulations are performed in PyBullet [39], a Python-based physics simulation engine, on a computer with Ubuntu 22.04 with Intel i7-12800 HX CPU, with 16GB RAM and NVIDIA 3070 Ti GPU. For the class function $\mathcal{K}$ in the CBF inequality, we chose $\kappa(B) = \gamma B$, where $\gamma = 1$. A perception boundary is also considered both in simulations and experiments to model the sensor's obstacle detection limits. Thus, only after entering the perception boundary, the obstacle is observable and the C3BF-QP is activated. All simulations and hardware experiments can be viewed on the project webpage [2].

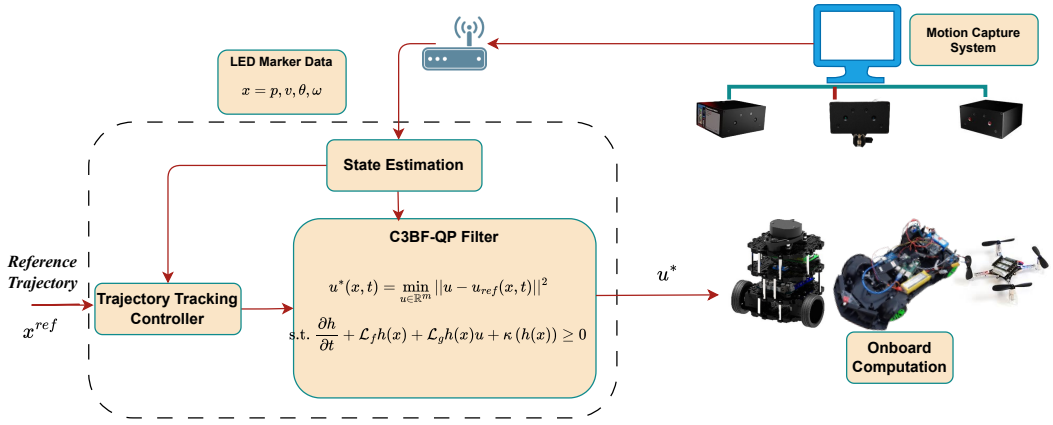


Figure 6.7: Experimental Setup for TurtleBot Burger, Stoch-Jeep and Crazyflie-2.1.

6.5.1 Unicycle Model

We consider the reference control inputs to be a simple PD controller. We chose constant target velocities to verify the C3BF-QP. However, the reference controller can be replaced by any trajectory tracking controller like the Stanley controller [68]. The QP yields the optimal accelerations, which are then applied to the robot.

Simulation Results: All the simulations were performed on TurtleBot3 UGV in PyBullet. We consider different scenarios with different initial poses and velocities of both the vehicle and the obstacle. Different scenarios include interaction with static and dynamic obstacles that result in turning, braking, reversing, and overtaking.

Hardware Results: The experiments were performed on TurtleBot3 Burger (Fig. 6.6), a two-wheeled differential drive type platform, to demonstrate the efficacy of C3BF on Unicycle models. The global position of the car, as well as the obstacle(s),

[2].https://tayalmanan28.github.io/CollisionConeCBF/

is measured using the PhaseSpace™ motion capture system with a tracking frequency of 960 Hz. The 2 LED markers are placed in front and rear of the bot to estimate its state (p, v, θ, ω). The control loop is operated at a frequency of 100 Hz. The maximum speed of the bot is 0.22m/s and maximum rotational velocity is 2.84 rad/s. The average radius of the obstacle for these experiments is 0.11m. The experimental setup is shown in Fig. 6.7.

Experiments are performed for the cases with single and multiple static and dynamic obstacles as shown in Fig 6.8, Fig 6.11a. The corresponding results are shown in Figs. 6.14-6.13. As is observed from Figs. 6.14a-6.13a, the robot successfully evades the obstacles in the reported trials. The corresponding command inputs and the resultant linear velocities are shown in Figs. 6.13c-6.14c and Figs. 6.13b-6.14b, respectfully. These results provide controlled-setting evidence of the proposed scheme under external motion-capture state estimates; they do not establish robustness to onboard perception or state-estimation errors.

6.5.2 Bicycle Model

We have extended and validated our C3BF algorithm for the bicycle model (6.2), which is a good approximation of actual car dynamics under the assumption of small lateral acceleration ($\leq 0.5\mu g$, μ is the friction co-efficient) [67] [120]. The linear acceleration reference control a_{ref} was obtained from a PD controller tracking the desired velocity, and the steering reference β_{ref} was obtained from the Stanley controller.

Simulation Results: All the simulations were performed on the bicycle model in PyBullet. We consider different scenarios with different initial poses and velocities of both the vehicle and the obstacle. The simulations show the interaction of bicycle with single and multiple static and dynamic obstacles.

Hardware Results: Corresponding to the cases tested in PyBullet simulation, experiments were performed on Stoch-Jeep (Fig. 6.6), a rear wheel drive car $1/10th$ scaled model, which is powered by 11.1 V, 1800 mAh LIPO Battery. The steering actuation is handled by a servo motor, TowerPro MG995, and a DC motor actuates the rear wheels. The global position of the car, as well as the obstacle(s), is again measured using PhaseSpace™ motion capture system with a tracking frequency of 960 Hz. The 2 LED Markers are placed in front and rear of the car to estimate the state of the car (p, v, θ, ω). The control loop is operated at a frequency of 100 Hz. The maximum speed of the bot is 0.2m/s and maximum steering angle it can achieve is 1 rad. The average radius of the obstacle for these experiments is 0.11m. Fig 6.7 shows the experimental setup.

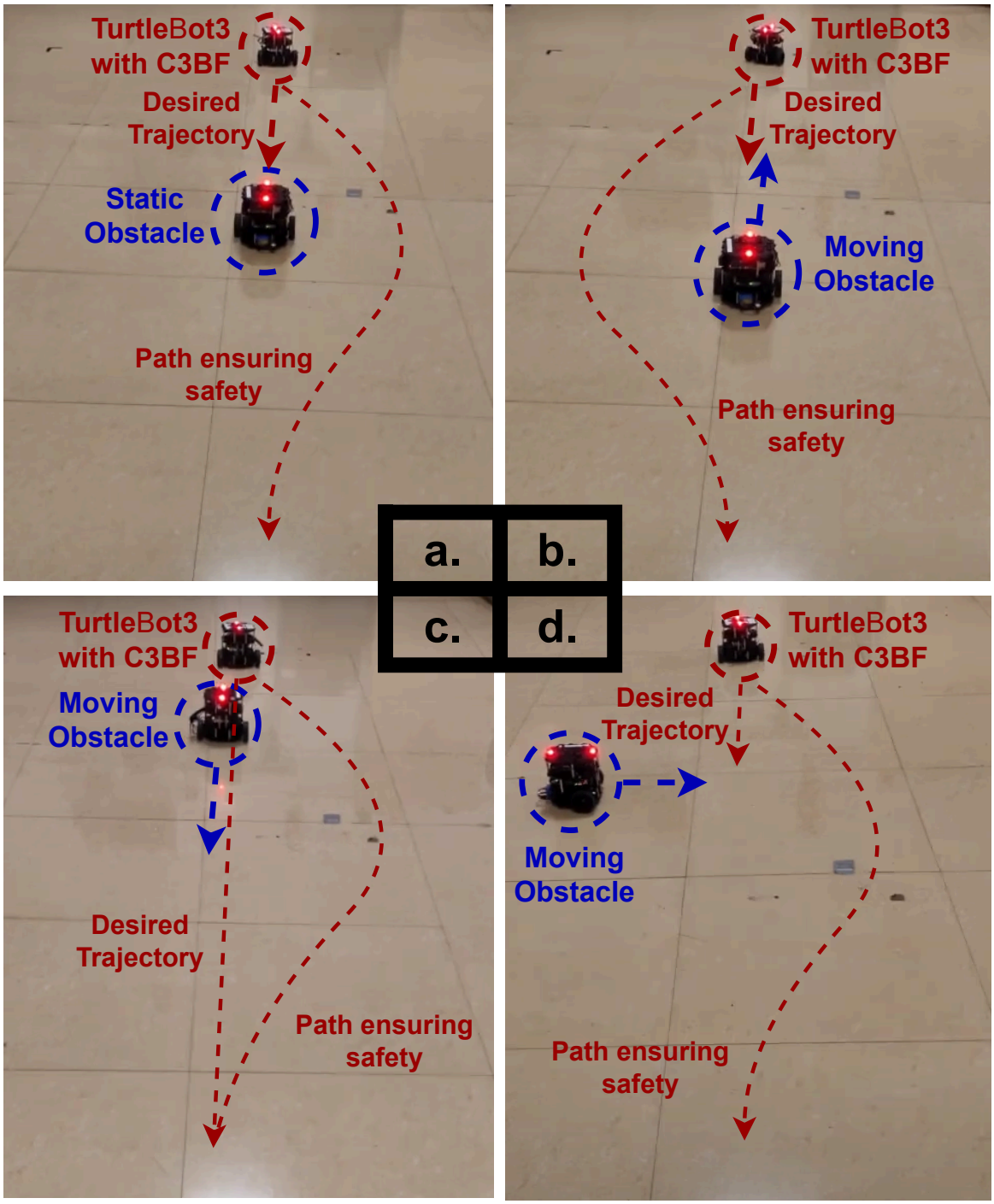


Figure 6.8: Interaction of TurtleBot3 (Unicycle Model) with (a) Static Obstacle; (b) Dynamic Obstacle (Velocity towards the agent); (c) Dynamic Obstacle (Overtaking); (d) Dynamic Obstacle (Velocity perpendicular to & towards the agent).

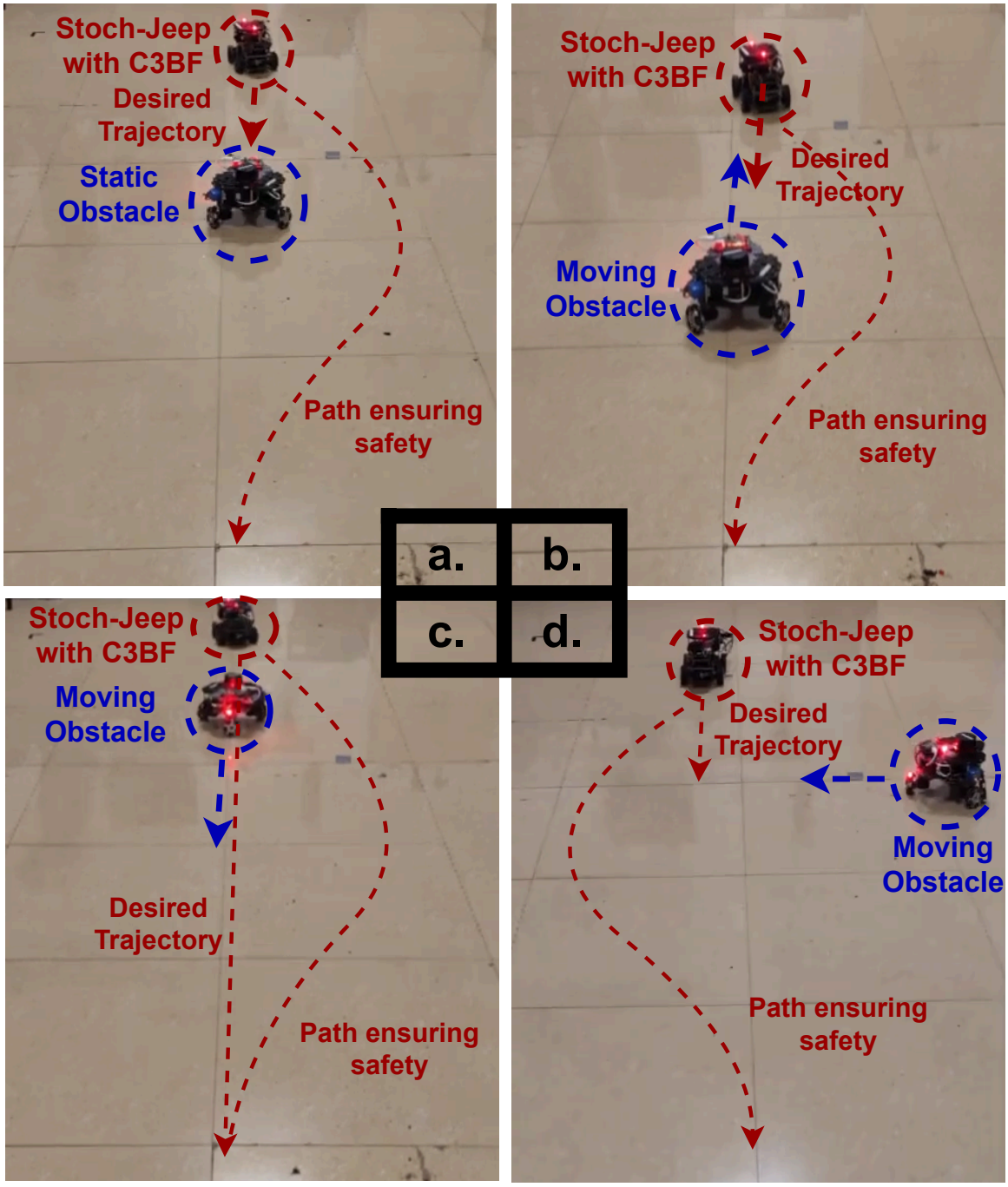


Figure 6.9: Interaction of Stoch-Jeep (Bicycle Model) with (a) Static Obstacle; (b) Dynamic Obstacle (Velocity towards the agent); (c) Dynamic Obstacle (Overtaking); (d) Dynamic Obstacle (Velocity perpendicular to & towards the agent).

Experiments are performed for the cases with single and multiple static and dynamic obstacles as shown in Fig 6.9, Fig 6.11b. The corresponding results are shown in Figs. 6.15-6.16. As is observed from Figs. 6.15a-6.16a, the robot successfully evades the obstacles in the reported trials. The corresponding command inputs and resultant linear velocities are shown in Figs. 6.15c-6.16c and Figs. 6.15b-6.16b, respectively. These controlled experiments validate the C3BF filter under the stated model and external localization, rather than a generic multi-agent perception setting.

6.5.3 Quadrotor Model

We have validated the C3BF-QP-based controller on quadrotors for the 3D and Projection CBF cases. Again, the PD controller is used as a reference controller to track the desired path, and the deployed safety controller2.11. We chose constant target velocities to verify the C3BF-QP.

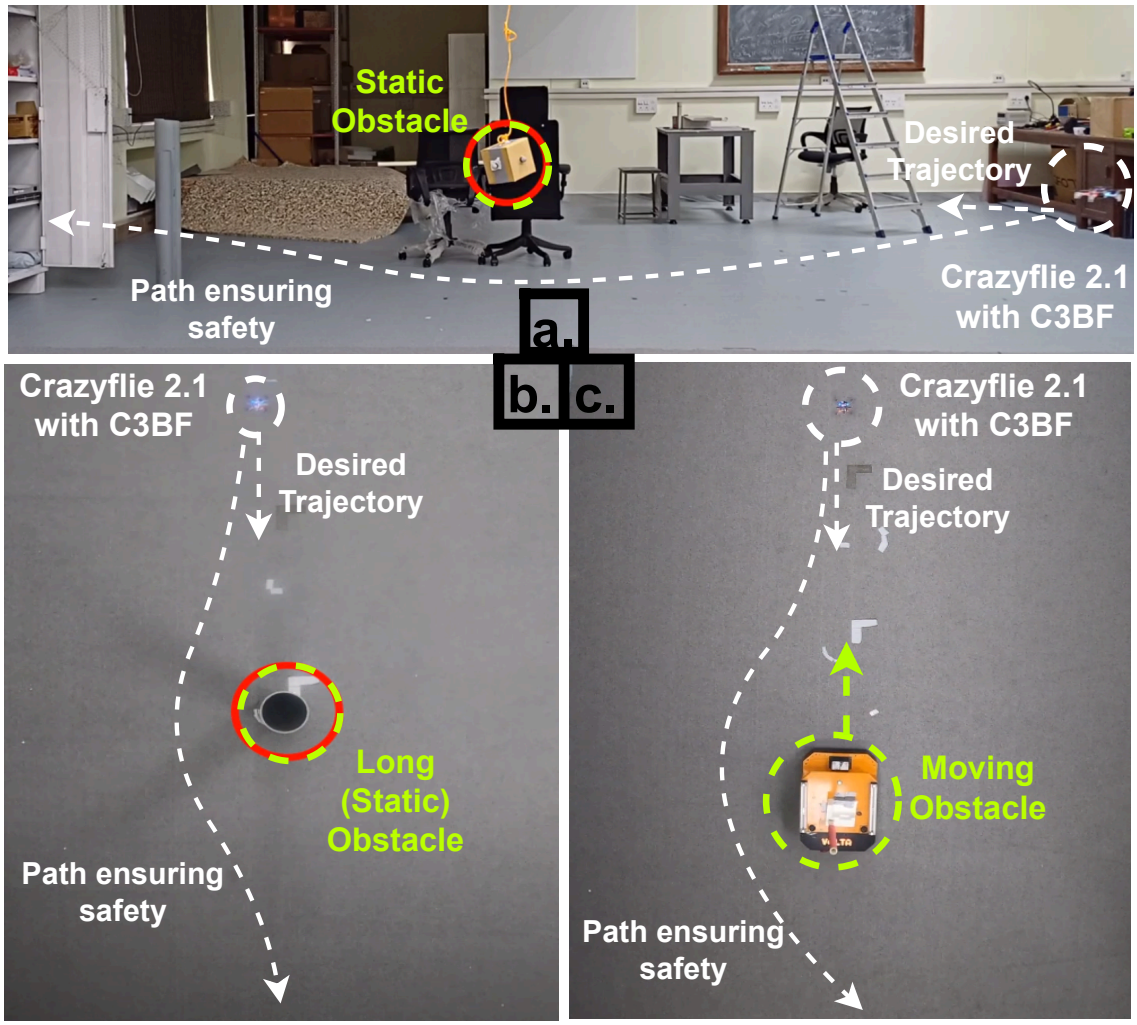


Figure 6.10: Interaction of Crazyflie 2.1 (Quadrotor) with (a) Static Obstacle; (b) Long (Static) Obstacle; (c) Dynamic Obstacle (Velocity towards the agent).

Simulation Results: The simulations were conducted using the multi-drone environment [114] on Pybullet. The simulations show the interaction of the quadrotor with static obstacles (3D case), moving obstacles (3D case) and elongated obstacles (Projection case). It also shows the comparison of trajectories of the quadrotor when following C3BF and HO-CBF. We observed that the HO-CBF could not avoid a high-speed approaching obstacle. In addition, it is not able to properly avoid the longer obstacles in the projection CBF case. The comparison between the thrusts (sum of the inputs) of the C3BF and HO-CBF based controllers is shown in Fig. 6.12

Hardware Results: The experiments were conducted using the Bitcraze™ Crazyflie 2.1 aerial drone to demonstrate the efficacy of the C3BF-QP controller framework. The mass (m) of the drone is 27gms. The global position of the drone and the obstacle is measured using Qualisys™ Miqus M3 motion capture system. Passive markers are placed on the drone to facilitate tracking with motion capture system. Further, for the drone, the global position obtained from the motion capture system is fused with the onboard IMU data via the Extended Kalman Filter to get the filtered state. The control commands are generated by an off-board computer and transmitted to the drone via a radio link. The control loop is operated at a frequency of 100 Hz. Communication with the drone is facilitated through the Crazyflie Python library [18]. The Crazyflie 2.1 can achieve a maximum thrust of 60gms. The average radius of obstacles considered in these experiments is 0.1m. The experimental setup is shown in Fig. 6.7. Experiments are performed for the cases with a single static obstacle, multiple static obstacles, and moving obstacles as shown in Fig 6.10, Fig 6.11c, 6.11d. The corresponds results are shown in Figs. 6.17,6.18. As is observed from Figs. 6.17a,6.18a, the quadrotor is able to successfully evade the obstacles. The corresponding command inputs and the resultant linear velocities are shown in Figs. 6.17c,6.18c and Figs. 6.17b,6.18b, respectfully. Hence, the experimental results verify the efficacy of the proposed scheme for obstacle avoidance.

> *Remark* 6.5.1. *As the quadrotor passes over the top of an obstacle, a ground effect is generated, which renders the drone unstable and leads to crashes. Hence, the obstacle avoidance simulations where the quadrotor passes over the obstacle could not be replicated in experiments.*

6.6 Chapter Summary

This chapter introduced Collision Cone Control Barrier Functions (C3BFs), a family of analytical CBFs that encode velocity-level collision avoidance constraints by embedding the collision cone condition into the barrier function framework. We developed C3BF formulations for unicycle, bicycle, and quadrotor models and demon-

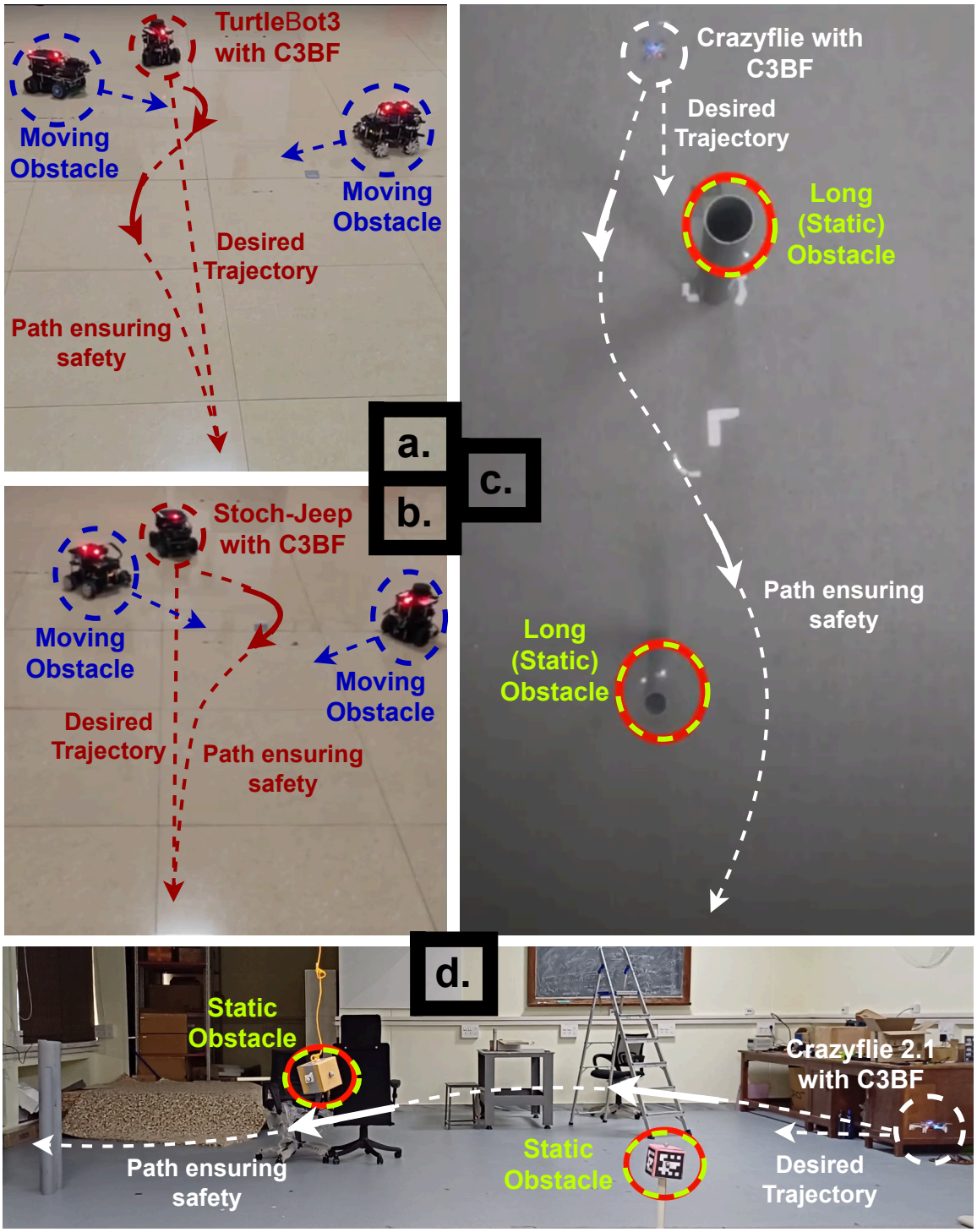


Figure 6.11: Robustness in scenarios with multiple obstacles: (a) TurtleBot3 (Unicycle Model) [Dynamic Obstacles]; (b) Stoch-Jeep (Bicycle Model) [Dynamic Obstacles]; (c) Quadrotor (Crazyflie 2.1) - Long (Static) Obstacles; (d) Quadrotor (Crazyflie 2.1) - Spherical (Static) Obstacles

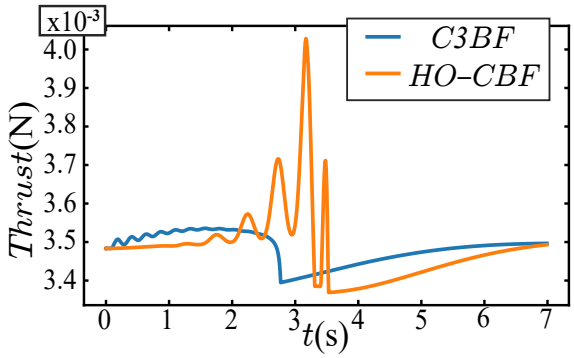
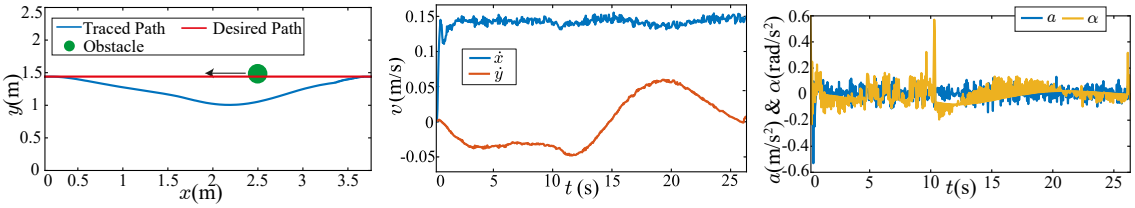
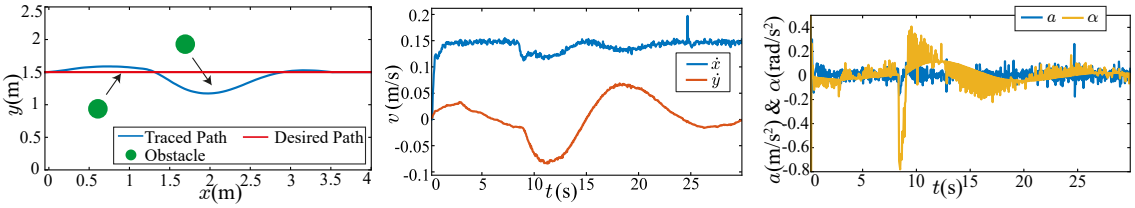


Figure 6.12: Comparison of thrusts between Collision Cone CBF and Higher Order CBF



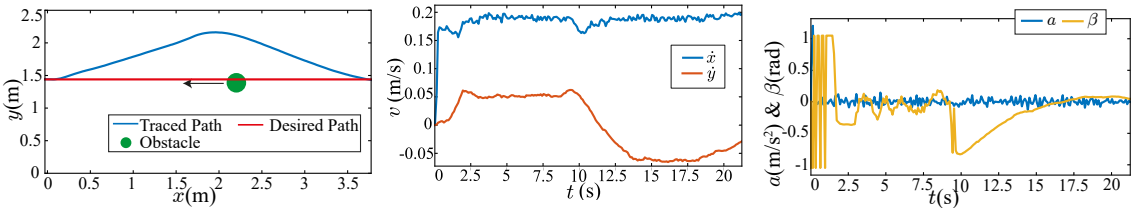
(a) (b) (c)

Figure 6.13: Experiments with a dynamic obstacle (unicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs.



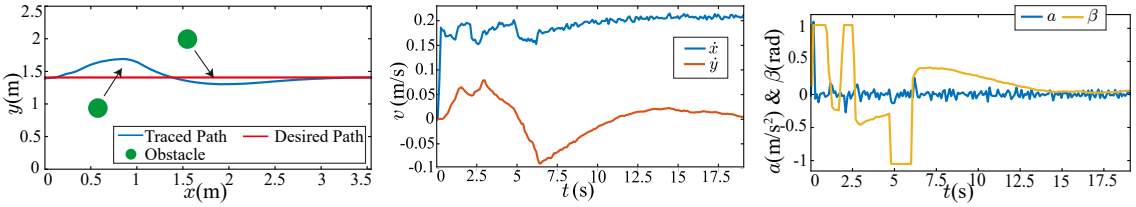
(a) (b) (c)

Figure 6.14: Experiments with multiple obstacles (unicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs.



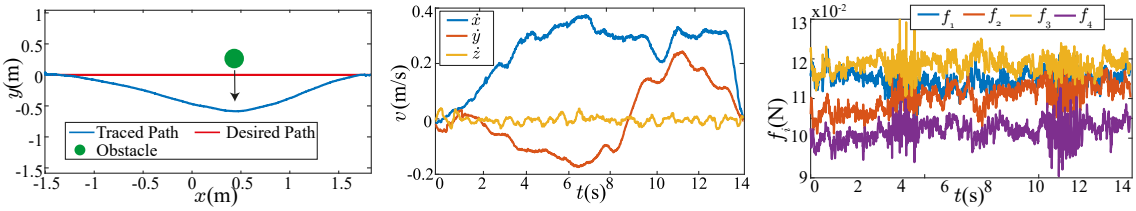
(a) (b) (c)

Figure 6.15: Experiments with dynamic obstacles (bicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs.



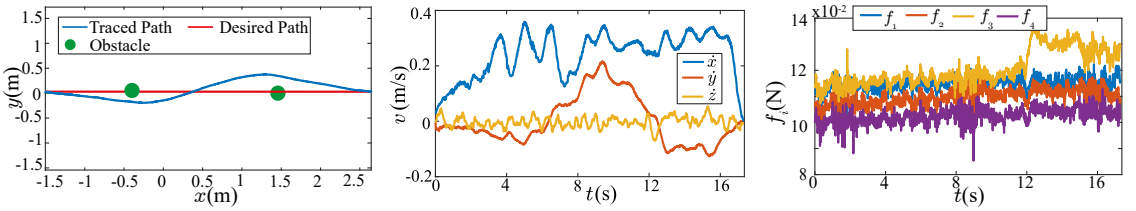
(a) (b) (c)

Figure 6.16: Experiments with multiple obstacles (bicycle): a) Traced Path, Evolution of b) linear velocity, c) control inputs.



(a) (b) (c)

Figure 6.17: Experiments with a dynamic obstacle (quadrotor): a) Traced Path, Evolution of b) linear velocity, c) control inputs.



(a) (b) (c)

Figure 6.18: Experiments with multiple obstacles (quadrotor): a) Traced Path, Evolution of b) linear velocity, c) control inputs.

strated their effectiveness for avoiding both static and dynamic obstacles through extensive simulation and hardware experiments. The extension to polygonal obstacles via PolyC2BFs further broadened the applicability to cluttered environments.

While C3BFs provide elegant, closed-form safety filters with real-time performance, they rely on several assumptions that limit their applicability to more complex scenarios. First, the barrier functions must be *manually designed* for each robot model and obstacle geometry, requiring significant domain expertise that does not transfer across platforms. Second, the approach assumes *accurate knowledge* of the system dynamics and obstacle states, which is rarely available in unstructured environments. Third, the framework does not account for *stochastic disturbances* such as sensor noise or wind gusts, meaning that safety guarantees may be violated when the true system deviates from the nominal model. Finally, scaling to high-dimensional systems or complex safe-set geometries becomes increasingly difficult, as analytical barrier construction requires geometric insight that does not generalize.

These limitations motivate a complementary, data-driven approach. In the next chapter, rather than prescribing the barrier function analytically, we learn it from offline reinforcement-learning value functions. The resulting Value-guided Offline CBF (V-OCBF) framework inherits the temporal look-ahead of reachability analysis without requiring known dynamics or hand-crafted geometry, producing barrier certificates that encode long-horizon safety directly from recorded data.

Chapter 7

Value-Guided Offline Neural CBF Synthesis

7.1 Related Work

7.1.1 Safe Offline RL

Safe RL has traditionally been studied in the online setting, where most methods rely on Lagrangian-based constrained optimization [35, 169, 147]. These approaches regulate safety through cost penalties, treating constraints as soft and therefore allowing non-zero violation risk. CPO [3] provides theoretical guarantees during updates through a trust-region mechanism, but still cannot ensure strict safety throughout online learning. These limitations motivate shifting toward safe offline RL , where policies are synthesized from static data to avoid unsafe exploration. Among early offline-safe RL methods, CPQ [188] assigns large costs to unsafe or out-of-distribution actions, but this can distort the value function and reduce generalization [92]. COptiDICE [89], building on DICE objectives [88], inherits the difficulties of residual-gradient learning [11]. More recent work integrates safety considerations into Decision Transformer [29] or Diffuser models [76], enabling sequence-modeling-based safety [101, 96, 204]. These sequence-model architectures are computationally expensive and challenging to scale. Building on the offline safety–performance perspective developed in this chapter, subsequent work by the thesis author extends this direction to offline reinforcement learning [166] and develops Safe Flow Q-Learning with reachability-based flow policies [167]. Overall, while offline RL mitigates unsafe online interaction, existing methods primarily enforce soft constraints and often struggle when unsafe transitions are underrepresented or missing from the dataset.

7.1.2 Offline Neural CBFs

Complementary to RL-based strategies, several recent methods aim to learn Control Barrier Functions (CBFs) directly from offline demonstrations. The approach in [132] enforces differentiable barrier conditions on expert trajectories but does not account for out-of-distribution (OOD) states, limiting generalization beyond demonstrated data. The iDBF framework [25] mitigates unsafe generalization by filtering low-likelihood states and actions using a Behavior Cloned policy, although this reliance on BC likelihood restricts coverage when demonstrations do not span the full safe set. Conservative CBFs [151] incorporate uncertainty-aware penalties to avoid optimistic predictions on OOD states, but this often yields overly cautious certificates that substantially underestimate the true safe region. While these approaches represent important progress, they fundamentally rely on the empirical demonstration distribution and do not explicitly model how system dynamics may evolve toward unsafe states. Consequently, they tend to produce conservative CBFs that underapproximate the true safe set.

7.1.3 Positioning of Our Work

V-OCBF differs from safe offline RL methods [89, 101, 96, 204], which rely on soft constraints, by enforcing a learned barrier condition as a hard constraint in a QP-based safety filter. The resulting one-step safety statement is conditional on the finite-difference barrier condition holding for the realized dynamics; the present analysis does not propagate explicit learned-dynamics or CBVF approximation-error bounds. Unlike offline neural CBF approaches [132, 25, 151] that remain tied to the demonstration distribution, V-OCBF employs a model-free value recursion that models future safety evolution and estimates a safe-set expansion beyond observed states. Overall, our method combines offline value learning with CBF-based control to obtain an offline-learned, actuation-aware neural CBF without requiring online rollouts.

7.2 Contributions of this Chapter

This chapter makes the following contributions:

1. We introduce *V-OCBF*, a framework that learns Neural Control Barrier Functions from offline data by using value functions as surrogate reachability approximations. The value recursion models how system dynamics evolve toward

or away from unsafe states, enabling the barrier to expand beyond the support of the demonstration distribution.

2. We formulate an *expectile-based value learning* objective (inspired by Implicit Q-Learning) that avoids the need for explicit policy evaluation on out-of-distribution actions, making the approach fully compatible with offline data.

3. We enforce the learned barrier condition as a *hard QP constraint*. This provides conditional one-step safety wherever the finite-difference barrier condition holds for the realized dynamics; explicit bounds on learned-model and CBVF approximation errors are not derived in this chapter.

4. We conduct extensive experiments across continuous-control benchmarks, demonstrating that V-OCBF produces *significantly larger safe sets* and *higher task performance* compared to existing offline CBF methods (NCBF, iDBF, Conservative CBF) and safe offline RL baselines (CPQ, COptiDICE, CDT, FISOR).

The results presented in this chapter are based on [168]. Additional material is available on the project webpage[1], and the accompanying code is available online[2].

7.3 Background and Problem Setup

We consider a control-affine nonlinear dynamical system defined by the state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, the control input $u(t) \in \mathbb{U} \subseteq \mathbb{R}^m$, and governed by the following dynamics:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{7.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous functions. We are given a set $\mathcal{C} \subseteq \mathcal{X}$ that represents the *safe states* for the system and a failure set $\mathcal{F} \subseteq \mathcal{X}$ that represents the set of unsafe states for the system (e.g., obstacles for an autonomous ground robot). The system is controlled by a Lipschitz continuous control policy $\pi : \mathbb{R}^n \to \mathbb{R}^m$. Our focus lies in ensuring the safety of this dynamical system, which is formally defined as follows:

Definition 7.3.1 (Safety)**.** A dynamical system is considered safe if the set, $\mathcal{C} \subseteq \mathcal{X} \subseteq \mathbb{R}^n$, is positively invariant under the control policy, π, i.e, $x(0) \in \mathcal{C}, u(t) = \pi(x(t)) \implies x(t) \in \mathcal{C},\ \forall\ t \geq 0$.

[1] https://tayalmanan28.github.io/v-ocbf/
[2] https://github.com/tayalmanan28/v-ocbf

Since, $\mathcal{F} \subseteq \mathcal{X} \setminus \mathcal{C}$, it can be trivially shown that $x(t) \in \mathcal{C} \implies x(t) \notin \mathcal{F} \; \forall \; t \geq 0$. Using this premise, we define the main objective of this paper:

Objective 7.1. Our objective is to synthesize a safe policy $\pi_{\text{safe}} : [t, T) \times \mathcal{X} \to \mathbb{U}$ such that the resulting closed-loop system satisfies the positive invariance property specified in Definition (7.3.1).

7.3.1 Control Barrier Functions

Control Barrier Functions [7, 8] are widely used to synthesize control policies with positive invariance guarantees, thereby ensuring system safety. The initial step in constructing a Control Barrier Function (CBF) involves defining a continuously differentiable function $B : \mathcal{X} \to \mathbb{R}$, where the *super-level set* of B corresponds to the safe region $\mathcal{C}$. This leads to the following representation:

$$\mathcal{C} = \{x \in \mathcal{X} : B(x) \geq 0\}, \quad \mathcal{X} \setminus \mathcal{C} = \{x \in \mathcal{X} : B(x) < 0\}. \tag{7.2}$$

The interior and boundary of $\mathcal{C}$ are further specified as:

$$\text{Int}(\mathcal{C}) = \{x \in \mathcal{X} : B(x) > 0\}, \quad \partial\mathcal{C} = \{x \in \mathcal{X} : B(x) = 0\}. \tag{7.3}$$

The function B qualifies as a valid Control Barrier Function if it satisfies the following definition:

Definition 7.3.2 ([8])**.** Given a control-affine system $\dot{x} = f(x) + g(x)u$, the set $\mathcal{C}$ defined by (7.2), with $\frac{\partial B}{\partial x}(x) \neq 0$ for all $x \in \partial\mathcal{C}$, the function B is called the Control Barrier Function (CBF) defined on the set $\mathcal{X}$, if there exists an extended *class-*$\mathcal{K}$ function κ such that for all $x \in \mathcal{X}$:

$$\max_{u \in \mathbb{U}} \left[\underbrace{\mathcal{L}_f B(x) + \mathcal{L}_g B(x) u}_{\dot{B}(x,u)} + \kappa\left(B(x)\right) \right] \geq 0, \tag{7.4}$$

where $\mathcal{L}_f B(x) = \frac{\partial B}{\partial x} f(x)$ and $\mathcal{L}_g B(x) = \frac{\partial B}{\partial x} g(x)$ are the Lie derivatives and n is the dimension of the system.

As established in [8], any Lipschitz continuous control law $\pi(x)$ that satisfies the condition $\dot{B} + \kappa(B) \geq 0$ guarantees the system's safety when $x(0) \in \mathcal{C}$. Additionally, if the initial state $x(0)$ lies outside $\mathcal{C}$, this condition ensures asymptotic convergence to the safe set $\mathcal{C}$.

While CBFs provide a principled framework to guarantee safety, their practical deployment is hindered by the lack of general methods for constructing valid barrier functions. As a result, practitioners typically resort to handcrafted or domain-specific CBFs, which can yield overly conservative safe sets. In the presence of control bounds, a nominal CBF may conflict with feasibility requirements, causing the corresponding CBF-QP to become infeasible.

7.3.2 Control Barrier Value Function (CBVF)

To overcome the limitations inherent in classical CBF formulations, [33] proposed the *Control Barrier Value Function* (CBVF), which integrates Control Barrier Functions with Hamilton–Jacobi (HJ) Reachability [12]. We begin by encoding the safety specification using a Lipschitz continuous function $\ell : \mathbb{R}^n \to \mathbb{R}$, where the failure set is defined as $\mathcal{F} := \{x \in \mathcal{X} \mid \ell(x) < 0\}$. Under this construction, a CBVF $B : \mathcal{X} \to \mathbb{R}$ is defined as the viscosity solution of the following Hamilton–Jacobi–Bellman Variational Inequality (HJB-VI):

$$\min\left\{ \max_{u\in\mathbb{U}}(\nabla B(x)\cdot(f(x)+g(x)u)),\ \ell(x)-B(x) \right\} = 0, \tag{7.5}$$

with boundary condition $B(x)|_{t=0} = \ell(x)|_{t=0}$. The resulting value function induces a forward-invariant safe set $\mathcal{C} := \{x \in \mathcal{X} \mid B(x) \geq 0\}$ and ensures that admissible controls $u \in \mathbb{U}$ satisfy the following Lie-derivative condition:

$$\max_{u\in\mathbb{U}} \left[L_f B(x) + L_g B(x)u + \kappa(B(x))\right] \geq 0. \tag{7.6}$$

Safe Controller Synthesis using CBVFs: Quite often, we have a reference control policy, $\pi_{ref}(x)$, designed to meet the performance requirements of the system. However, such controllers often lack safety guarantees. To ensure the system meets its safety requirements while preserving performance, the reference controller must be minimally adjusted to incorporate safety constraints. This adjustment can be accomplished using the Control Barrier Value Function-based Quadratic Program (CBVF-QP), described as follows:

$$\begin{aligned} \pi_{\text{safe}}(x) = \min_{u\in\mathbb{U}\subseteq\mathbb{R}^m} &\|u - \pi_{ref}\|^2 \\ \text{s.t. } &\mathcal{L}_f B(x) + \mathcal{L}_g B(x)u + \kappa\left(B(x)\right) \geq 0. \end{aligned} \tag{7.7}$$

The CBVF-QP framework facilitates the synthesis of a provably safe control policy, $\pi_{\text{safe}}(x)$, while staying close to the reference controller to preserve system performance.

Challenges in CBVF Synthesis: Traditional approaches compute Control Barrier Value Functions using grid-based HJ reachability methods [107], which are fundamentally limited by the curse of dimensionality. Recent efforts have attempted to overcome these issues by learning CBVFs through online reinforcement learning [145]; however, such approaches require extensive online interaction, rendering them unsuitable for safety-critical systems. Furthermore, existing neural CBF methodologies [2, 198, 165] typically assume access to accurate system dynamics, an assumption that does not hold for many real-world platforms. These limitations motivate a shift toward using offline demonstrations, either sourced from public datasets [100, 149] or collected in controlled settings where safety can be guaranteed. Building on this premise, we refine Objective 7.1 as follows:

Objective 7.2. Our objective is to synthesize a Control Barrier Value Function $B : \mathcal{X} \to \mathbb{R}$ directly from an offline dataset of demonstrations $\mathcal{D}$, such that the resulting CBVF-QP controller π_{safe} in (7.7) satisfies the positive invariance property specified in Definition 7.3.1.

7.4 Methodology

Having introduced the CBVF formulation in the previous section, we now describe a practical methodology for synthesizing a valid Control Barrier Value Function and thereby the safe controller from offline demonstrations $\mathcal{D}$. Our objective is to construct a data-driven approximation of the viscosity solution of the HJB-VI (7.5) without relying on known system dynamics or online interaction. The key idea is to re-interpret it through a finite-difference barrier recursion compatible with demonstration data, and to use this recursion to learn a value-guided barrier function that inherits forward invariance.

7.4.1 Finite-Difference Barrier Synthesis

To approximate the CBVF using trajectories in $\mathcal{D}$, we consider a finite difference recursive version of equation (7.5). Given a trajectory $\{x_t, u_t\}_{t=0}^{N}$, we define the finite-difference barrier update

$$B(x_t) = \min\left\{\ell(x_t),\ \max_{u_t} B(x_{t+1})\right\}, \qquad \forall t \in \{0, 1, 2, \dots\}, \tag{7.8}$$

where, $x_t = x(t)$, $u_t = u(t)$ and $x_{t+1} = x(t + \Delta t)$, along with the boundary condition $B(x_0) = \ell(x_0)$. This recursion has two significant advantages. First, it enables us

to learn a barrier directly from data without requiring f and g. Second, under mild regularity assumptions, (7.8) preserves the forward invariance property of the CBVF. The derivation of the recursive equation is provided in Appendix D.1.1. Intuitively, the recursion encodes the principle that a state is safe if and only if its immediate successor is safe or it lies outside the unsafe set as specified by $\ell(x)$.

To represent this barrier function, we parameterize a neural network $B^{\mathcal{D}}_{\psi}(x)$ with ψ utilizing the universal approximation property. However, directly solving for the recursion (7.8) can lead to degenerate solutions. For instance, setting $B^{\mathcal{D}}_{\psi}(x) = c$ for a sufficiently small constant c satisfies (7.8) but clearly does not satisfy the CBVF conditions in (7.5). This pathology is analogous to the non-contractive behavior of undiscounted value iteration in MDPs. Following the approach in [56], we incorporate a discounted finite-difference loss to avoid such trivial solutions:

$$\mathcal{L}_{B^{\mathcal{D}}}(\psi) = \mathbb{E}_{(x,u,x')\sim\mathcal{D}}\Big[\Big((1-\gamma)\,\ell(x) + \gamma \min\big\{\ell(x),\, B^{\mathcal{D}}_{\psi}(x')\big\} - B^{\mathcal{D}}_{\psi}(x)\Big)^2\Big], \tag{7.9}$$

where $x = x_t$, $x' = x_{t+1}$ and $u = u_t$ and $\gamma \to 1$. This discounted recursion ensures contraction, promotes stable learning, and prevents the network from collapsing to uniformly unsafe or uniformly safe solutions. To avoid evaluating barrier targets at out-of-distribution actions, we remove the maximization over actions from the loss in (7.9) and use only the demonstrated action in each transition. While this prevents unsupported queries, the resulting estimate reflects the safety profile of the behaviour policy that generated the data. Consequently, this produces a behaviour-induced barrier, which is typically sub-optimal because it ignores other admissible actions that could yield larger safe-set estimates.

7.4.2 Avoiding Out-of-Distribution Actions in Offline Learning

The naïve regression objective in (7.9) fits $B^{\mathcal{D}}_{\psi}$ to the mean of the demonstrated next-state targets, but this corresponds to the behavior-induced barrier and yields overly conservative safe sets. Ideally, if we assume unlimited capacity and no sampling error, the optimal parameters should satisfy, $B(x) \approx \mathbb{E}_u\big[\min\big\{\ell(x), \max_u B(x')\big\}\big]$. However, such unconstrained maximization can result in actions that are never observed in the dataset.

Since offline data only provides information about those actions selected by the behavior policy, evaluating values (or barrier targets) using unsupported actions can distort learning because the corresponding transitions are not grounded in the dataset. Subsequently, motivated by the insights of Implicit Q-Learning (IQL) [85], we approximate the maximization over admissible actions by using expectile regression, which

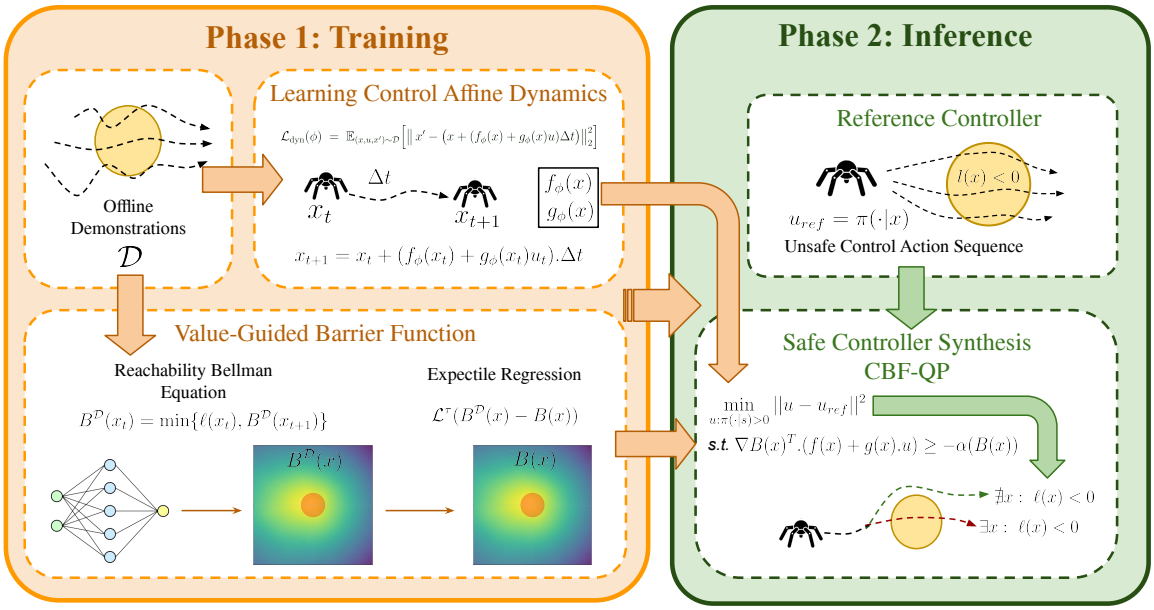


Figure 7.1: **Framework Overview:** *(Left)*: We learn value guided barrier function with reachability based bellman equation for learning optimal safe region and apply expectile regression for OOD case handling, when learning from Offline Dataset. *(Right)*: Inferencing with CBF-QP using learned barrier function as valid CBF to rollout safe step-wise actions, with any reference controller.

enables us to capture the highest admissible barrier values supported by the data while never evaluating B on unseen (x, u) pairs. This allows us to perform a principled value-style backup over the dataset control action support $\mathcal{U}_{\mathcal{D}} = \{\, u \mid (x, u, x') \in \mathcal{D} \,\}$ without extrapolating to unsafe or unobserved actions. IQL shows that expectile regression produces a value function that reflects the values induced by the behavior policy without requiring an explicit behavior model. This prevents the learning target from being influenced by actions that lie outside the dataset support, while still capturing the highest feasible values supported by the demonstrations.

Following this principle, we estimate a CBVF, B_θ, with θ as the Neural Network parameters, that reflects the safety values implied by demonstrated actions. Formally, we minimize the expectile loss

$$\mathcal{L}_B(\theta) \;=\; \mathbb{E}_{(x,u,x')\sim\mathcal{D}}\big[\mathcal{L}^\tau\big(B^{\mathcal{D}}_\psi(x) - B_\theta(x)\big)\big], \tag{7.10}$$

where $\mathcal{L}^\tau(y) = |\tau - \mathbf{1}(y < 0)|\, y^2$ is the τ-expectile loss used in [85]. Intuitively, a higher expectile level τ places greater weight on underestimation errors than overestimation errors, pushing B_θ toward the upper envelope of safety values supported by the dataset. Thus, τ controls how aggressively the learned barrier emphasizes high, data-supported safety values without extrapolating to unseen actions. The barrier function thus obtained, B_θ, is our proposed *Value-guided Offline Control Barrier Function* (V-OCBF). The complete learning procedure for $B^{\mathcal{D}}_\psi$ and the V-OCBF B_θ

is summarized in Algorithm 7 and can be visually seen in Fig. 7.1.

Algorithm 7 Learning V-OCBF from Offline Demonstrations

Require: Inputs: Offline dataset $\mathcal{D}$, expectile level τ, batch size m, epochs N, learning rates η_ψ, η_θ, discount factor γ
Ensure: Outputs: Learned CBVF $B_\psi^{\mathcal{D}}$ and V-OCBF B_θ

```
Initialize ψ, θ and optimizers
for epoch = 1 to N do
    for minibatch B ⊂ D do
        (A) B_ψ^D update
        L_{B^D} ← (x, u, x', γ, ψ) with (x, u, x') ∼ B        ▷ From Eq. 7.9
        ψ ← AdamStep(ψ, ∇_ψ L_{B^D})
        (B) form support targets
        For each unique x ∈ D gather support U_D from D
        (C) B-network losses
        L_B ← (x, U_D, τ, B_ψ^D, θ)                           ▷ From Eq. 7.10
        θ ← AdamStep(θ, ∇_θ L_B)
    end for
end for
Return B_θ
```

7.4.3 Controller Synthesis via Learned Dynamics

The learned barrier function B is subsequently employed to fulfill the primary goal of synthesizing a safe policy 7.1 using the CBVF-QP formulation in (7.7). Solving this QP necessitates the evaluation of the Lie derivatives $\mathcal{L}_f B$ and $\mathcal{L}_g B$, both of which rely on the underlying control-affine system dynamics. In our offline-only setting, the true dynamics are unavailable; therefore, we construct a neural network–based surrogate model to approximate the underlying transition dynamics of the form:

$$x_{t+1} = x_t + \big(f_\phi(x_t) + g_\phi(x_t)u_t\big)\,\Delta t, \tag{7.11}$$

which enables computation of the required derivatives and supports safe policy synthesis. The model parameters ϕ are trained using one-step transitions from the offline dataset $\mathcal{D}$ by minimizing the prediction loss:

$$\mathcal{L}_{\text{dyn}}(\phi) = \mathbb{E}_{(x,u,x')\sim\mathcal{D}}\Big[\,\|\,x' - \big(x + (f_\phi(x) + g_\phi(x)u)\Delta t\big)\,\|^2\Big], \tag{7.12}$$

implemented as a minibatch MSE objective. The full training procedure is summarized in Algorithm 8.

Importantly, the learned dynamics model is *not* used when learning the barrier function. Incorporating it into the CBVF learning stage would require evaluating

terms involving (f_ϕ, g_ϕ) under actions outside the dataset-supported set $\mathcal{U}_\mathcal{D}$, thereby violating the action constraints critical for preventing value underestimation in the offline regime. Hence, using learned dynamics during CBVF training would allow the network to extrapolate into unsupported regions of the action space, defeating the purpose of the OOD-aware barrier learning objective described earlier.

In contrast, at *inference* time, the learned dynamics serve a different role: they enable the evaluation of Lie derivatives needed to solve the CBVF-QP ((7.7)). Specifically, for any query state x, we compute

$$\mathcal{L}_f B(x) = \nabla_x B_\theta(x)^\top f_\phi(x), \qquad \mathcal{L}_g B(x) = \nabla_x B_\theta(x)^\top g_\phi(x). \tag{7.13}$$

These quantities allow the QP in (7.7) to be solved for the safe control action u_{safe}, completing the pipeline for constructing a safety-certified controller purely from offline demonstrations.

In Section 7.5.2, we provide an analysis demonstrating the limitations of using learned dynamics inside the CBVF learning loop, further reinforcing the necessity of restricting their use to the inference stage only.

Algorithm 8 Controller Synthesis

Require: Inputs: Learned barrier B_θ, offline dataset $\mathcal{D} = \{(x, u, x')\}$, batch size m, epochs N, learning rate η_ϕ, step Δt, reference controller u_{ref}, function κ, query state $\hat{x}$

Ensure: Outputs: Learned affine dynamics f_ϕ, g_ϕ and safe controller $u_{\text{safe}}(\hat{x})$

Initialize ϕ and optimizer
for epoch $= 1$ to N **do**
  for minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
    $L_{\text{dyn}}(\phi) \leftarrow (x, u, x', \Delta t, \phi)$ ▷ From Eq. 7.12
    $\phi \leftarrow \text{AdamStep}(\phi, \nabla_\phi L_{\text{dyn}})$
  end for
end for

Inference at state $\hat{x}$
$L_f B(\hat{x}) \leftarrow (\nabla_x B_\theta(\hat{x}), f_\phi(\hat{x}))$ and $L_g B(\hat{x}) \leftarrow (\nabla_x B_\theta(\hat{x}), g_\phi(\hat{x}))$ ▷ From Eq. 7.13
$u_{safe} \leftarrow \texttt{QP-Solver}\big(L_f B(\hat{x}),\, L_g B(\hat{x}),\, u_{\text{ref}}(\hat{x}),\, B_\theta(\hat{x}), \kappa\big)$ ▷ From Eq. 7.7
return safe controller $u_{safe}(\hat{x})$

7.5 Experiments

The experiments are designed to evaluate: (i) the safety and performance of V-OCBF relative to constrained offline RL and neural CBF baselines on systems with unknown

dynamics, (*ii*) the advantages of value-guided barriers over behavior-policy–induced barriers, (*iii*) the robustness of the resulting QP controller under external disturbances, and (*iv*) the effectiveness of V-OCBF compared to a CBVF synthesized using learned dynamics.

Baselines: We compare V-OCBF against a diverse set of constrained offline learning and CBF-based methods. For constrained offline learning, we include **Behavior Cloning (BC)**, **BEAR-Lag** (Lagrangian constraint version of [86]), **COptiDICE** [89], and **FISOR** [204] which enforce safety indirectly via soft constraints on policy optimization or behavior imitation. For CBF-based approaches, we evaluate **Neural Control Barrier Function (NCBF)** [132], *Conservative Control Barrier Function (CCBF)* [151], and *In-Distribution Barrier Function (iDBF)* [25], which synthesize explicit safety filters from offline data but often yield conservative safe sets. In contrast, V-OCBF learns a *value-guided* barrier function from offline demonstrations that accounts for future unsafe interactions, producing a hard, state-wise safety filter with larger safe set coverage.

Evaluation Metrics: We evaluate all methods based on (i) *safety*, measured as the total number of safety violations incurred before episode termination, and (ii) *performance*, measured via the cumulative episode rewards. These metrics allow us to assess the trade-off between strict safety enforcement and task performance across different offline RL and CBF-based approaches.

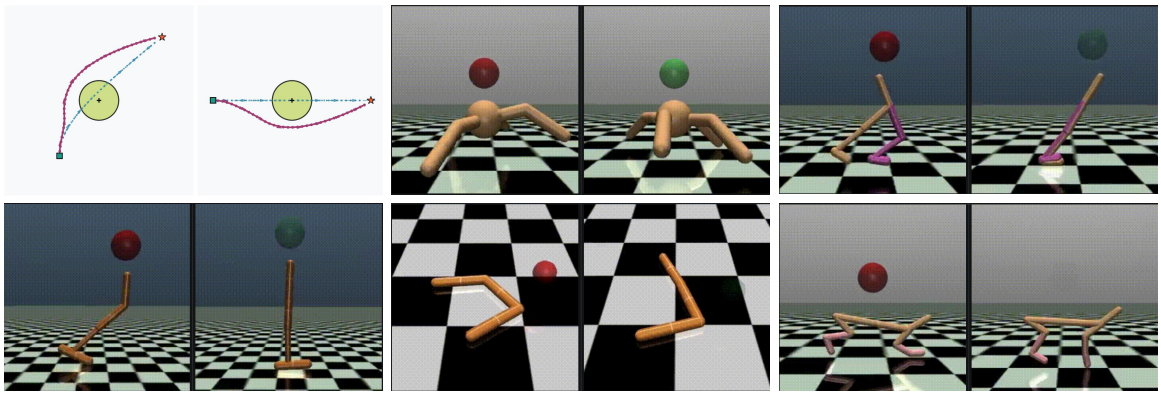


Figure 7.2: **Illustrations of the test environments:** (*Top-Left*) environment illustrates Autonomous Ground Vehicle with an obstacle and a goal point. (*MuJoCo Env*) **Red** sphere denotes unsafe state.

7.5.1 Experimental Case Studies

To perform a holistic performance analysis of our proposed approach, we apply V-OCBF in conjunction with Behavior Cloning (BC) as the nominal (reference) con-

troller for all the different environments which are supposed to assess varying objectives. Below we list all the environments that we use:

- **Autonomous Ground Vehicle (AGV) Collision Avoidance:** In our first experiment, we examine a 3-dimensional collision avoidance problem involving an autonomous ground vehicle governed by Dubins' car dynamics [50]. The objective is to ensure safety by avoiding a static obstacle while navigating through a bounded environment. Further details on the system dynamics, state space bounds, and experimental setup are provided in Appendix D.2.1.
- **MuJoCo Safety Gymnasium:** We next evaluate our framework on Safety Gymnasium environments [77]. Specifically, we evaluate the V-OCBF-based QP (equation 7.7) on high-dimensional MuJoCo tasks like Hopper, Swimmer, Half Cheetah, Walker2D and Ant. The objective in each environment is to maximize reward while keeping the agent velocity below the velocity thresholds. We keep the reward and safety-violation metrics identical to the Safety-Gymnasium definitions and use the standard DSRL dataset for safe offline RL [100]. To evaluate our method against baselines, we randomly sampled 500 initial states for each environment, respectively, the results for which can be referred to from Figure 7.3.

7.5.2 Results

Effectiveness in Co-optimizing Safety and Performance

We begin by evaluating all methods on the AGV Collision Avoidance task, which provides a clear setting to study how different approaches balance safety and performance. The results in Table 7.1 highlight notable differences in how offline RL and CBF-based methods handle this trade-off.

Offline RL baselines such as BC, BEAR-Lag, and COptiDICE achieve relatively low safety rates. BC tends to reproduce unsafe behaviors from the dataset, while BEAR-Lag and COptiDICE try to account for safety but remain limited because they operate with soft constraint formulations. Their lower reward and safety scores indicate that they struggle to balance both safety and performance objectives.

In contrast, methods that incorporate a CBF-QP layer, such as BC+NCBF, BC+iDBF, and BC+CCBF, achieve much higher safety rates. The QP ensures that unsafe actions are filtered out, even if the nominal controller is imperfect. However, the performance of these approaches still depends heavily on the quality of the learned

Table 7.1: AGV Collision Avoidance Experiment: Percentage Safe Episodes, Mean Episode Reward and Safe Set Volume (refer Appendix D.2.1) across different methods. Evaluated over 500 episodes and 5 seed values.

Method	**Safe Episodes (%)**	**Episode Reward**	**Safe Set Volume (%)**
BC	48.92 ± 1.69	20.45 ± 1.84	42.51
BEAR-Lag [86]	65.12 ± 0.24	13.85 ± 0.81	58.21
COptiDICE [89]	68.91 ± 0.32	15.33 ± 0.67	62.32
BC+NCBF [132]	92.48 ± 0.60	44.61 ± 2.58	81.92
BC+iDBF [25]	92.87 ± 0.73	48.23 ± 2.01	83.32
BC+CCBF [151]	93.56 ± 0.56	49.66 ± 2.34	90.94
FISOR [204]	95.78 ± 0.2	52.33 ± 0.93	90.14
BC+**V-OCBF** (Ours)	**98.28 ± 0.54**	**54.93 ± 0.46**	**92.57**

barrier. FISOR performs better than the other offline RL baselines because it explicitly expands the feasible safe region before optimizing for performance. However, due to lack of explicit safety filtering, it leads to lesser safety rates than our proposed method, due to the impending learning errors in the computation of feasible region. This also highlights the importance of QP based safety filtering scheme for achieving better safety. Overall, *V-OCBF achieves the strongest results across all metrics.*

To further analyze scalability, we extend the evaluation to MuJoCo Safety Gymnasium environments (Hopper, Half-Cheetah, Ant, Swimmer, and Walker2D), with unknown dynamic models. The results in Figure 7.3 demonstrate that V-OCBF again achieves the lowest safety violation rates across all tasks. Notably, the method maintains near-zero violations on while preserving satisfactory reward levels compared to BC and outperforming iDBF, NCBF, and CCBF. These neural CBF baselines degrade sharply in higher dimensions: NCBF suffers from optimization difficulties, while iDBF often enforces overly restrictive boundaries that suppress task performance. FISOR again remains competitive but does not match the safety consistency of V-OCBF.

Overall, the experiments provide strong empirical evidence that V-OCBF effectively co-optimizes safety and performance, scaling from low-dimensional AGV dynamics to complex MuJoCo systems. The method consistently outperforms existing offline RL and neural CBF baselines in terms of safety while maintaining competitive reward, highlighting its suitability for offline settings where both strict safety and reliable performance are required.

Effect of learned dynamics

We additionally examine how using the learned dynamics influences both CBVF learning and QP-based control on the AGV environment, where ground-truth dynamics

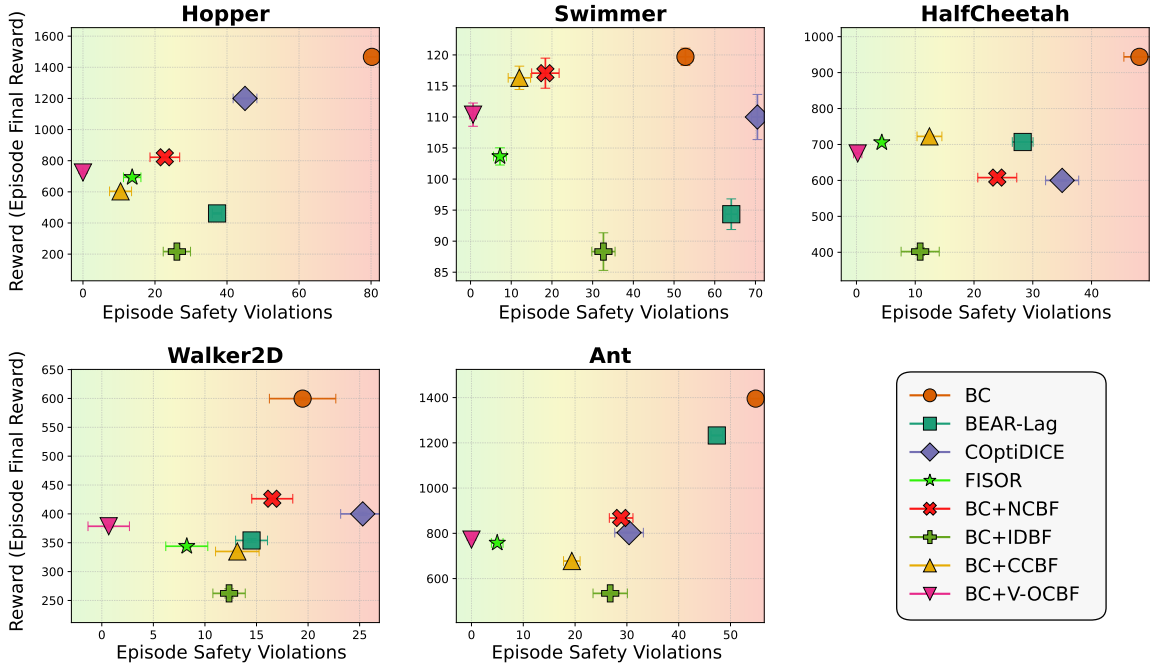


Figure 7.3: Results plot for all the Mujoco based Safety Gymnasium [77] environments. Points towards left (←) are more safe than those on right (→). Evaluated over 500 episodes and 5 seed values.

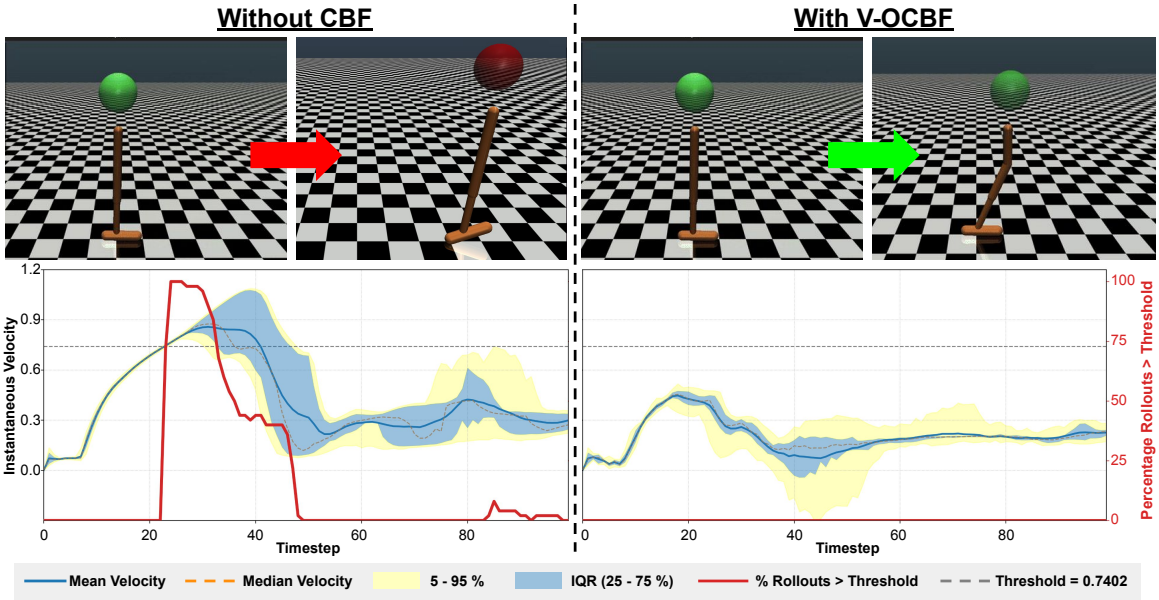


Figure 7.4: **Qualitative comparison of Hopper rollouts with and without V-OCBF.** *(Top):* When using V-OCBF, the agent adapts its gait to keep its forward velocity within the allowable safety threshold, demonstrating how the learned barrier actively regulates unsafe accelerations. *(Bottom):* Instantaneous velocities across episodes for both methods. The left axis reports the velocity profile, while the right axis shows the fraction of episodes that violate the safety threshold at each timestep. V-OCBF substantially reduces violation frequency, illustrating its ability to enforce safety constraints during deployment. Additional qualitative results for other environments are provided in Appendix D.2.2.

are available for comparison. Table 7.2 reports safety and reward for three configurations: (i) V-OCBF with a QP evaluated using learned dynamics, (ii) V-OCBF with a QP evaluated using true dynamics, and (iii) a neural CBVF trained using learned dynamics and QP-based controller evaluated on the same dynamics.

Across all metrics, using learned dynamics at inference time yields performance that is close to the known-dynamics case, indicating that the controller synthesis using QP is robust to moderate model error. In contrast, using learned dynamics during CBVF training leads to a noticeable drop in both safety and reward. This aligns with our earlier discussion in section 7.4.3: learning the barrier through a model introduces distributional mismatch because the learned dynamics may generate actions outside the dataset-supported set $\mathcal{U}_{\mathcal{D}}$, yielding suboptimal targets for B. These findings reinforce our design choice, dynamics should be used only at inference to compute the lie derivatives in the QP, while barrier learning itself should avoid dependence on a learned model.

Table 7.2: Use of Learned Dynamics at the time of Training v/s Inference (Mean over 500 episode rollouts).

Method	**Safe Episodes (%)**	**Episode Reward**
BC+**V-OCBF** (QP with Learned Dynamics)	98.28 ± 0.54	54.93 ± 0.46
BC+**V-OCBF** (QP with Known Dynamics)	99.52 ± 0.58	55.91 ± 0.39
BC+**CBVF from Learned Dynamics**	94.64 ± 0.65	46.40 ± 0.02

7.5.3 Ablation Studies

To further evaluate the robustness of V-OCBF, we conduct ablation studies that examine: (i) the sensitivity of the learned safe set to the size of the barrier network, (ii) the effect of disturbances on the safety of QP based controller, and (iii) the dependence of the learned barrier on the expectile parameter τ. Across these experiments, V-OCBF exhibits stable safety performance over a broad range of architectural and hyperparameter choices, remaining resilient even under perturbations to the control actions. These findings suggest that the framework generalises reliably beyond the specific tasks considered and is well-suited for settings where safety certification must be carried out under imperfect system knowledge. The full results and visualisations for these studies are presented in Appendix D.3.

7.6 Chapter Summary

This chapter presented V-OCBF, a framework that bridges offline reinforcement learning and safety certification by using learned value functions as surrogate reachability approximations to guide the synthesis of Neural CBFs. Unlike prior offline CBF methods that remain tied to the demonstration distribution, V-OCBF uses model-free value recursion to estimate a safe-set expansion beyond observed states. The resulting barrier conditions are imposed as hard constraints by the QP controller, while the value-function shaping aligns the learned safe set with task-relevant regions of the state space. Safety remains conditional on those learned barrier conditions holding for the realized dynamics. Extensive experiments demonstrated larger learned safe sets and higher task performance compared to the evaluated offline CBF and safe RL baselines.

While V-OCBF successfully integrates safety with performance considerations, it operates within the *offline RL* paradigm, inheriting its limitations: the quality of the learned barrier depends on the coverage of the offline dataset, and the approach does not exploit knowledge of the system dynamics. The framework addresses single-agent settings and does not provide mechanisms for multi-agent coordination.

Scope of the Guarantees. We further clarify the precise strength of the safety statement in this chapter, since two of its ingredients—the dynamics model used inside the QP and the CBVF surrogate shaping the barrier—are themselves *learned*. The barrier conditions here are enforced on sampled states via the finite-difference condition, and no explicit bound on the approximation errors of these learned components currently enters the safety statement; consequently, the statement is best understood as *conditional*: trajectories remain safe wherever the finite-difference barrier condition holds for the realized dynamics. Because the CBVF is shaped by value recursion over $\mathcal{D}$, both transition fidelity and long-horizon coverage of safety-relevant regions limit how far beyond demonstrations the learned safe-set expansion can be trusted. The existing ablations in Appendix D.3 examine network size, control perturbations, the expectile parameter, and aggregate learned-dynamics error; they provide empirical evidence about robustness in those tested settings, but do not systematically vary dataset coverage or rollout horizon. A more rigorous treatment would propagate explicit error bounds into an inflated barrier margin, e.g., via Lipschitz constants of the learned dynamics model, dataset-coverage certificates, or by conformally calibrating the residual violation rate on closed-loop rollouts in the spirit of Chapter 4. Developing such error-aware certificates for value-guided barrier synthesis is a principal direction of ongoing work.

In the next chapter, we take a complementary approach by developing a *physics-informed* machine learning framework that directly solves the Hamilton–Jacobi–Bellman PDE to co-optimize safety and performance. By exploiting the known dynamics structure through physics-informed training and combining it with conformal prediction for formal verification, we obtain a framework that is both more principled (grounded in optimal control theory) and more scalable (applicable to high-dimensional systems without offline datasets).

Part III

Bridging Safety with Performance

Bridging the gap between safety and optimality, this part uses physics-informed neural networks to jointly solve Hamilton–Jacobi–Bellman equations for safe and optimal control, scaling from single-agent to cooperative multi-agent settings.

Chapter 8

Co-optimizing Safety and Performance via Physics-Informed ML

8.1 Related Works

As discussed in Chapter 1, existing approaches to safe control broadly fall into three categories: constrained reinforcement learning (CRL) [5, 3], safety filtering [69, 8, 20, 176], and online optimization-based methods such as MPC [58, 61] and MPPI [185, 148]. CRL methods only softly impose safety constraints by minimizing cumulative violations rather than enforcing them at all times. Safety filters, while providing hard constraint satisfaction, operate independently of the performance objective and can lead to myopic decisions. Online optimization methods can integrate both objectives but are computationally expensive for real-time deployment, especially with nonlinear dynamics and nonconvex constraints. None of these approaches jointly provide formal safety certificates and performance optimality guarantees.

State-Constrained Optimal Control. A principled approach to co-optimizing safety and performance is the *state-constrained optimal control problem (SC-OCP)*, where safety is encoded as a hard state constraint and performance through a cost function. Characterizing SC-OCP solutions requires restrictive controllability conditions [146]. To overcome this, [4] proposed an epigraph-based reformulation that characterizes the SC-OCP value function via dynamic programming, yielding a Hamilton-Jacobi-Bellman PDE (HJB-PDE). The value function and optimal policy are then recovered from this epigraph. Building on the epigraph-based framework developed

in this chapter, subsequent work by the thesis author extends the same perspective to flow matching for synthesizing safe and performant policies from offline data [164]. However, grid-based numerical methods for solving the HJB-PDE suffer from the curse of dimensionality [107, 178], and the epigraph formulation further increases the state dimension by one. Speed-up techniques such as [34] restrict the class of admissible systems, leaving the problem of solving the HJB-PDE for general nonlinear systems open.

Physics-Informed Learning for HJ PDEs. Physics-informed neural networks (PINNs), introduced in Chapter 2, offer a mesh-free alternative for solving PDEs by embedding the governing equations into the training loss [128, 127]. DeepReach [13] adapts this idea to Hamilton-Jacobi reachability, training neural networks to approximate value functions satisfying HJB or HJI PDEs. Follow-up works [94, 142, 178, 143] have extended this to various safety-critical settings. However, DeepReach and its variants focus exclusively on safety (i.e., computing the backward reachable tube) without incorporating performance objectives, leading to overly conservative policies. Moreover, these learned solutions lack formal correctness guarantees, leaving verification of the resulting policies as an open problem.

Verification of Learned Value Functions. While Chapter 4 developed conformal prediction-based verification for neural control barrier functions, verifying learned *value functions* in the optimal control setting poses distinct challenges: one must certify not only safety (constraint satisfaction) but also bound the performance gap between the learned and true optimal policies. Existing works on conformal prediction for control [95, 94] have focused on providing probabilistic safety certificates for learned reachability solutions. In this chapter, we apply conformal prediction in a dual role: to verify safety of the learned value function and to quantify performance degradation, providing high-confidence bounds on both without requiring access to the ground-truth solution.

Robust Formulations. The preceding discussion assumes deterministic dynamics. In practice, systems face disturbances and model uncertainty, motivating robust formulations. HJI reachability analysis [108, 55] provides exact robust solutions through differential game formulations but inherits the same computational limitations. We extend the physics-informed learning framework to the robust setting, formulating a Robust SC-OCP (RSC-OCP) and solving the resulting HJI variational inequality with neural networks, while generalizing the conformal verification procedures to the disturbance-aware case.

8.2 Contributions of this Chapter

The main contributions of this chapter are as follows:

- We propose a Physics-Informed Machine Learning (PIML) framework that approximates the auxiliary value function of the epigraph-reformulated SC-OCP by training a neural network to satisfy the HJB-PDE, enabling co-optimization of safety and performance for high-dimensional nonlinear systems.
- We develop a conformal prediction-based safety verification procedure that provides high-confidence probabilistic safety guarantees for the learned value function, along with a conformal prediction-based performance quantification method that bounds the gap between the learned and optimal policies.
- We extend the framework to the robust setting by formulating an RSC-OCP with bounded disturbances, training robust value functions via the Hamilton-Jacobi-Isaacs Variational Inequality, and generalizing the conformal prediction procedures to account for worst-case disturbances.
- Across three case studies of increasing complexity (2D boat navigation, 8D pursuer-evader tracking, and 20D multi-agent coordination), we demonstrate the effectiveness of the proposed method. We further validate the robust framework through hardware experiments on TurtleBot3 platforms, demonstrating effective sim-to-real transfer enabled by the disturbance-aware formulation.

The results presented in this chapter are based on [160, 162].

8.3 Problem Setup

Consider a nonlinear dynamical system characterized by the state $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and control input $u \in \mathcal{U} \subseteq \mathbb{R}^m$, governed by the dynamics $\dot{x}(t) = f(x(t), u(t), d(t))$, where the function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is locally Lipschitz continuous. In this work, we assume that the dynamics model f is known; however, it can also be learned from data if unavailable.

We are given a failure set $\mathcal{F} \subseteq \mathcal{X}$ that represents the set of unsafe states for the system (e.g., obstacles for an autonomous ground robot). The system's performance is quantified by the cost function $C(t, x, \mathbf{u}, \mathbf{d})$, given by:

$$C(t, x(t), \mathbf{u}, \mathbf{d}) = \int_{s=t}^{T} l(x(s))\, ds + \phi(x(T)), \tag{8.1}$$

where $l : \mathcal{X} \to \mathbb{R}_{\geq 0}$ and $\phi : \mathcal{X} \to \mathbb{R}_{\geq 0}$ are Lipschitz continuous and non-negative functions, representing the running cost over the time horizon $[t, T)$ and the terminal cost at time T, respectively. $\mathbf{u} : [t, T) \to \mathcal{U}$ is the control signal applied to the system and $\mathbf{d} : [t, T) \to \mathcal{D}$ is the disturbance of the system. We note that (8.1) is stated with a purely state-dependent running cost for clarity of exposition; the formulation—and the epigraph reformulation of Section 8.3.2—extend verbatim to running costs $l(x, u)$ that penalize control effort as in LQR-type objectives (indeed, one recovers the LQR problem by choosing $l(x, u) = x^\top Q x + u^\top R u$ and dropping the state constraint). The case studies in this chapter employ progress- and time-optimality costs without an explicit input-energy penalty; adding one changes only the definition of l and none of the developments that follow. Using this premise, we define the main objective of this paper:

Objective 8.1. We aim to synthesize an optimal policy $\pi^* : [t, T) \times \mathcal{X} \to \mathcal{U}$ that minimizes the cost function C while ensuring that the system remains outside the failure set $\mathcal{F}$ at all times.

If we do not have access to the system dynamics f, we further limit the scope of the problem to employing a model-based approach - first model the system of interest, then provide safety assurances on the system against any bounded model uncertainty $d \in \mathcal{D}$, where $\mathcal{D}$ is the set of admissible model uncertainty. We state the proposed safety assurance objective as follows:

Objective 8.2 ((Safety Assurance Under Uncertainty))**.** Given a failure set $\mathcal{F}$ and time horizon $[0, T]$, obtain a set $\mathcal{R} \subseteq X$, along with a state-feedback controller $\pi_{\mathcal{R}}$, such that under the condition that all realized model uncertainties are contained within $\mathcal{D}$, system f starts within $\mathcal{R}$ implies that f does not enter $\mathcal{F}$ for $[0, T]$ using controls from $\pi_{\mathcal{R}}$, or equivalently, $\mathcal{R} \subseteq S^*$.

8.3.1 Robust State-Constrained Optimal Control Problem

To achieve the stated objective, the first step is to encode the safety constraint via a function $g : \mathbb{R}^n \to \mathbb{R}$ such that, $\mathcal{F} := \{x \in \mathcal{X} \mid g(x) > 0\}$. Using these notations, the objective can be formulated as the following Robust State-Constrained Optimal Control Problem (RSC-OCP) to compute the value function V:

Problem 8.1 (Robust State-Constrained OCP)**.**

$$\begin{aligned} V(t, x(t)) = \min_{\mathbf{u}} \max_{\mathbf{d}} & \int_t^T l(x(s))ds + \phi(x(T)) \\ s.t.\ \dot{x} &= f(x, u, d), \\ g(x(s)) &\leq 0 \quad \forall s \in [t, T] \end{aligned} \tag{8.2}$$

This RSC-OCP enhances the system's performance by minimizing the cost, while maintaining system safety through the state constraint, $g(x) \leq 0$, ensuring that the system avoids the failure set, $\mathcal{F}$. Thus, the policy, π^*, derived from the solution of this RSC-OCP co-optimizes safety and performance.

What co-optimization means. Throughout this chapter, *co-optimization* denotes the synthesis of a single policy that simultaneously (i) respects the hard state constraint $g(x(s)) \leq 0$ along entire trajectories (*safety*) and (ii) minimizes the accumulated cost C (*performance*), rather than cascading a nominal performance controller with an external safety filter that overrides it. Efficacy is accordingly judged by two complementary metric families, both reported for every case study: *safety metrics*—the empirical safety (collision-free) rate over rollouts, together with the certified violation level δ obtained from the conformal verification procedures—and *performance metrics*—the mean task cost relative to the unconstrained optimum, and the computation time (offline training vs. online inference, Section 8.4.9).

8.3.2 Epigraph Reformulation

Directly solving the RSC-OCP in (8.2) presents significant challenges due to the presence of (hard) state constraints. To address this issue, we reformulate the problem in its epigraph form [22], which transforms the constrained optimization into a more tractable two-stage optimization problem. This reformulation allows us to efficiently obtain a solution to the RSC-OCP in (8.2). The resulting formulation is given by:

$$\begin{aligned} V(t, x(t)) = \min_{z \in \mathbb{R}^+} \ & z \\ \text{s.t. } \hat{V}(t, x, z) &\leq 0, \end{aligned} \tag{8.3}$$

where z is a non-negative auxiliary optimization variable, and $\hat{V}$ represents the auxiliary value function. Here, $\hat{V}$ is defined as [4]:

$$\begin{aligned} \hat{V}(t, x(t), z) = \min_{\mathbf{u}} \max_{\mathbf{d}} \max\{ & C(t, x(t), \mathbf{u}, \mathbf{d}) - z, \\ & \max_{s \in [t,T]} g(x(s))\}. \end{aligned} \tag{8.4}$$

Note that if $\hat{V}(t, x, z) < 0$, it implies that $g(x(s)) < 0$ for all $s \in [t, T]$. In other words, the system must be outside the failure set at all times; therefore, the system is guaranteed to be safe whenever $\hat{V}(t, x, z) < 0$.

In this reformulated problem, state constraints are effectively eliminated, enabling the use of dynamic programming to characterize the value function, as we explain later in this section. Intuitively, optimal z (z^*) can be thought of as the *minimum permissible cost* the policy can incur without compromising on safety. From Equation 8.3, it can be inferred that if $z > z^*$, the safety constraint dominates in the max term, resulting in a conservative policy. Conversely, if $z < z^*$, the performance objective takes precedence, leading to a potentially aggressive policy that might compromise safety.

Remark 8.3.1 (Why Train the Auxiliary Value Function Rather than V Directly?). One might consider learning the constrained value function V in (8.2) directly within the PINN framework. We deliberately avoid this route: the state-constrained HJ variational inequality satisfied by V carries the constraint $g \leq 0$ as an obstacle term, its viscosity solution is generally non-smooth across the constraint boundary, and residual-based training on such kinked solutions converges poorly in practice. The epigraph reformulation instead absorbs the constraint into the augmented state (x, z), producing an unconstrained HJB in n+1 dimensions that admits a continuous viscosity solution under mild assumptions ($\mathcal{A}_1'$–$\mathcal{A}_4'$ above); its interior residual is smooth and substantially better behaved for gradient-based training, and both quantities of interest—safety (via $\hat{V} \leq 0$) and the optimal cost level (via z^*)—are read off the same learned solution. The price is one additional network input dimension, which is negligible relative to the gain in trainability.

Furthermore, to facilitate solving the epigraph reformulation, z can be treated as a state variable, with its dynamics given by $\dot{z}(t) = -l(x(t))$. This implies that as the trajectory progresses over time, the minimum permissible cost, z, decreases by the step cost $l(x)$ at each time step. This allows us to define an augmented system that evolves according to the following dynamics:

$$\dot{\hat{x}} = \hat{f}(t, \hat{x}, u, d) := \begin{bmatrix} f(t, x, u, d) \\ -l(x) \end{bmatrix}, \tag{8.5}$$

where $\hat{x} := [x, z]^T$ represents the augmented state. With the augmented state representation and under assumptions A1–A4 of [4]:

$$(\mathcal{A}_1'): \begin{cases} (i)\ f(t,x,u,d) \in \mathbb{R}^d \text{ is continuous,} \\ \text{where } (t,x,u,d) \in [0,T]\times\mathbb{R}^d\times\mathcal{U}\times\mathcal{D}, \\ (ii)\ \exists L \geq 0,\ \forall x,y \in \mathbb{R}^d,\ \forall (u,d) \in \mathcal{U}\times\mathcal{D}, \\ \forall t,s \in [0,T],\ |f(t,x,u,d) - f(s,y,u,d)| \\ \leq L(|x-y| + |t-s|). \end{cases} \tag{8.6}$$

$$(\mathcal{A}_2'): \begin{cases} (i)\ \ell(t,x,u,d) \in \mathbb{R} \text{ is continuous,} \\ \text{where } (t,x,u,d) \in [0,T]\times\mathbb{R}^d\times\mathcal{U}\times\mathcal{D}, \\ (ii)\ \exists L \geq 0,\ \forall x,y \in \mathbb{R}^d,\ \forall (u,d) \in \mathcal{U}\times\mathcal{D}, \\ \forall t,s \in [0,T],\ |\ell(t,x,u,d) - \ell(s,y,u,d)| \\ \leq L(|x-y| + |t-s|). \end{cases} \tag{8.7}$$

$$(\mathcal{A}_3'): \quad \varphi : \mathbb{R}^d \to \mathbb{R} \text{ is Lipschitz continuous.} \tag{8.8}$$

The following convexity assumption on $\hat{f}$ will be considered:

$$(\mathcal{A}_4'): \quad \forall t,x,\beta,\ \hat{f}(t,x,u,d) \text{ is a convex set.} \tag{8.9}$$

The auxiliary value function $\hat{V}(t,x(t),z(t))$ is a unique continuous viscosity solution satisfying the following Hamilton–Jacobi–Isaacs Variational Inequality (HJI-VI):

$$\min\Big(-\partial_t \hat{V} - \min_{\mathbf{u}} \max_{\mathbf{d}} \langle \nabla_{\hat{x}} \hat{V}(t,\hat{x}),\ \hat{f}(\hat{x},u)\rangle,\ \hat{V} - g(x)\Big) = 0, \tag{8.10}$$

$\forall t \in [0,T)$ and $\hat{x} \in \mathcal{X}\times\mathbb{R}$, where $\langle\cdot,\cdot\rangle$ denotes the dot product of vectors. The boundary condition for the PDE is given by:

$$\hat{V}(T,\hat{x}) = \max\left(\phi(x(T)) - z, g(x)\right), \quad \hat{x} \in \mathcal{X}\times\mathbb{R}. \tag{8.11}$$

Note that by a slight abuse of notations, we have replaced the arguments x, z for $\hat{V}$ with the augmented state $\hat{x}$.

8.4 Safe and Optimal Control with Perfect Information and no Disturbances

With no disturbances and perfect information case we need to solve the following SC-OCP:

Problem 8.2 (State-Constrained OCP)**.**

$$\begin{aligned} V(t, x(t)) = \min_{\mathbf{u}} \int_t^T l(x(s))ds + \phi(x(T)) \\ s.t.\ \dot{x} = f(x, u), \\ g(x(s)) \leq 0 \quad \forall s \in [t, T] \end{aligned} \tag{8.12}$$

To solve the SC-OCP in Equation (8.12), we aim to compute the optimal value function V, which minimizes the cost while ensuring system safety. In this section, we outline a structured approach: first, we learn the auxiliary value function $\hat{V}$ using a physics-informed machine learning framework. Then, we apply a conformal prediction-based method to verify safety and correct for potential learning errors in $\hat{V}$. The final value function V is obtained from the safety-corrected $\hat{V}$ using the epigraph formulation in (8.3). Lastly, we assess the performance of V through a second conformal prediction procedure. Figure 8.1 gives an overview of the proposed approach. The following subsections provide a detailed explanation of each step, beginning with the methodology for learning $\hat{V}$.

Objective 8.3 ((Safety Assurance Under No Uncertainty))**.** Given a failure set $\mathcal{F}$ and time horizon $[0, T]$, obtain a set $\mathcal{R} \subseteq X$, along with a state-feedback controller $\pi_{\mathcal{R}}$, such that under the condition that all realized model uncertainties are contained within $\mathcal{D}$, system f starts within $\mathcal{R}$ implies that f does not enter $\mathcal{F}$ for $[0, T]$ using controls from $\pi_{\mathcal{R}}$, or equivalently, $\mathcal{R} \subseteq S^*$.

8.4.1 Training the Auxiliary Value Function ($\hat{V}$)

The auxiliary value function, $\hat{V}$, satisfies the HJB-PDE in Equation (8.10), as discussed in Section 8.3.2. Traditionally, numerical methods are used to solve the HJB-PDE over a grid representation of the state space [107, 135], where time and spatial derivatives are approximated numerically. While grid-based methods are accurate for low-dimensional problems, they struggle with the curse of dimensionality – their computational complexity increases exponentially with the number of states – limiting their use in high-dimensional systems. To address this, we adopt a physics-informed machine learning framework, inspired by [13], which has proven effective for high-dimensional reachability problems.

The solution of the HJB-PDE inherently evolves backward in time, as the value function at time t is determined by its value at $t + \Delta t$. To facilitate neural network training, we use a curriculum learning strategy, progressively expanding the time sampling interval from the terminal time $[T, T]$ to the full time horizon $[0, T]$. This

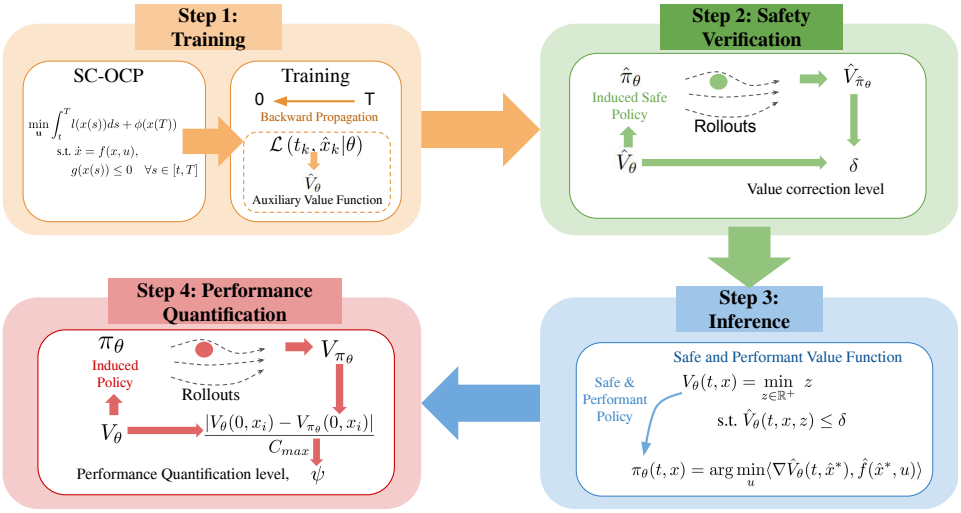


Figure 8.1: **Overview of the proposed approach**: The methodology is organized into four steps. The **first step** involves training the auxiliary value function, $\hat{V}_\theta$, using a physics-informed machine learning framework. The **second step** applies a conformal prediction approach for safety verification of the learned $\hat{V}_\theta$. In the **third step**, the final value function V_θ and the optimal safe and performant policy π_θ are inferred. The **fourth step** quantifies the performance of V_θ through a second conformal prediction procedure.

approach allows the neural network to first accurately learn the value function from the terminal boundary conditions, subsequently propagating the solution backward in time by leveraging the structure of the HJB-PDE.

Specifically, the auxiliary value function is approximated by a neural network, $\hat{V}_\theta$, where θ denotes the trainable parameters of the network. Training samples, $(t_k, x_k, z_k)_{k=1}^N$, are randomly drawn from the state space based on the curriculum training scheme. The proposed learning framework utilizes a loss function that enforces two primary objectives: (i) compliance with the PDE in (8.10), using the PDE residual error given by:

$$\begin{aligned}\mathcal{L}_{pde}\left(t_k, \hat{x}_k|\theta\right) = \| \min \Big\{ &-\partial_t \hat{V}_\theta\left(t_k, \hat{x}_k\right) - H(t_k, \hat{x}_k), \\ &\hat{V}_\theta\left(t_k, \hat{x}_k\right) - g\left(x_k\right) \Big\} \|,\end{aligned} \tag{8.13}$$

where $H(t, \hat{x}) = \min_{u \in \mathcal{U}} \langle \nabla \hat{V}_\theta(t, \hat{x}), \hat{f}(\hat{x}, u) \rangle$ and (ii) satisfaction of the boundary condition in (8.11), using boundary condition loss, given by:

$$\begin{aligned}\mathcal{L}_{bc}\left(t_k, \hat{x}_k|\theta\right) = \| &\max\left(\phi(x_k) - z_k, g(x_k)\right) - \\ &\hat{V}_\theta\left(t_k, \hat{x}_k\right) \| \mathbb{1}\left(t_k = T\right).\end{aligned} \tag{8.14}$$

These terms are balanced by a trade-off parameter λ, leading to the overall loss

Algorithm 9 Safety Verification using Conformal Prediction

Require: Inputs: Safe-state domain $\mathcal{S}$, calibration size N_s, confidence parameter β_s, target violation level ϵ_s, learned value $\hat{V}_\theta$, induced value $\hat{V}_{\hat{\pi}_\theta}$, number of candidate levels M

Ensure: Output: Calibrated safety-correction level δ

$D_0 \leftarrow$ Sample N_s IID states from $\mathcal{S}_{\delta=0}$
$\delta_0 \leftarrow \min_{\hat{x}_j \in D_0}\{\hat{V}_\theta(0,\hat{x}_j) : \hat{V}_{\hat{\pi}_\theta}(0,\hat{x}_j) \geq 0\}$
$\epsilon_0 \leftarrow$ (8.19) (using $\alpha_{\delta=0}$)
$\Delta \leftarrow$ Ordered list of M uniform samples from $[\delta_0, 0]$
for $i = 0, 1, \ldots, M-1$ **do**
    while $\epsilon_i \leq \epsilon_s$ **do**
        $\delta_i \leftarrow \Delta_i$
        Update α_{δ_i} from δ_i
        $\epsilon_i \leftarrow$ (8.19) (using α_{δ_i})
    end while
end for
return $\delta \leftarrow \delta_i$

function:

$$\mathcal{L}\left(t_k, \hat{x}_k|\theta\right) = \mathcal{L}_{pde}\left(t_k, \hat{x}_k|\theta\right) + \lambda \mathcal{L}_{bc}\left(t_k, \hat{x}_k|\theta\right) \tag{8.15}$$

Furthermore, we use the adaptive loss re-balancing scheme proposed in [181] to reduce the impact of λ on the learned value function. Minimizing the overall loss function provides a self-supervised learning mechanism to approximate the auxiliary value function.

8.4.2 Safety Verification

The learned auxiliary value function, $\hat{V}_\theta$, induces a policy, $\hat{\pi}_\theta$, that minimizes the Hamiltonian term $H(t, \hat{x})$ in the HJB-PDE. The policy is given by:

$$\hat{\pi}_\theta(t, \hat{x}) = \arg\min_{u \in \mathcal{U}} \langle \nabla \hat{V}_\theta(t, \hat{x}), \hat{f}(\hat{x}, u) \rangle. \tag{8.16}$$

The rollout cost corresponding to this policy is defined as:

$$\hat{V}_{\hat{\pi}_\theta}(t, \hat{x}) = \max\{C(t, x(t), \mathbf{u}) - z, \max_{s \in [t,T]} g(x(s))\}\Big|_{\mathbf{u}=\hat{\pi}_\theta} \tag{8.17}$$

Ideally, the rollout cost from a given state under $\hat{\pi}_\theta$ should match the value of the auxiliary value function at that state. However, due to learning inaccuracies, discrepancies can arise. This becomes critical when a state, $\hat{x}_i$, is deemed safe by the auxiliary value function ($\hat{V}_\theta(t, \hat{x}) \leq 0$) but is unsafe under the induced policy ($\hat{V}_{\hat{\pi}_\theta}(t, \hat{x}) > 0$). To address this, we introduce a uniform value function correction

margin, δ, which guarantees that the sub-δ level set of the auxiliary value function remains safe under the induced policy. Mathematically, the optimal δ (δ^*) can be expressed as:

$$\delta^* := \min_{\hat{x} \in \mathcal{X}} \{\hat{V}_\theta(0, \hat{x}) : \hat{V}_{\hat{\pi}_\theta}(0, \hat{x}) \geq 0\} \tag{8.18}$$

Intuitively, δ^* identifies the tightest level of the value function that separates safe states under $\hat{\pi}_\theta$ from unsafe ones. Hence, any initial state within the sub-δ^* level set is guaranteed to be safe under the induced policy, $\hat{\pi}^*_\theta$. However, calculating δ^* exactly requires infinitely many state-space points. To overcome this, we adopt a conformal-prediction-based approach to approximate δ^* using a finite number of samples, providing a probabilistic safety guarantee. The following theorem formalizes our approach:

Theorem 8.4.1 (Safety Verification Using Conformal Prediction)**.** *Let $\mathcal{S}_\delta$ be the set of states satisfying $\hat{V}_\theta(0, \hat{x}) \leq \delta$, and let $(0, \hat{x}_i)_{i=1,\dots,N_s}$ be N_s i.i.d. samples from $\mathcal{S}_\delta$. Define α_δ as the safety error rate among these N_s samples for a given δ level. Select a safety violation parameter $\epsilon_s \in (0, 1)$ and a confidence parameter $\beta_s \in (0, 1)$ such that:*

$$\sum_{i=0}^{l-1} \binom{N_s}{i} \epsilon_s^i (1 - \epsilon_s)^{N_s - i} \leq \beta_s, \tag{8.19}$$

where $l = \lfloor (N_s + 1)\alpha_\delta \rfloor$. Then, with the probability of at least $1 - \beta_s$, the following holds:

$$\mathbb{P}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s. \tag{8.20}$$

The proof is available in Appendix E.1.1. The safety error rate α_δ is defined as the fraction of samples satisfying $\hat{V}_\theta \leq \delta$ and $\hat{V}_{\hat{\pi}_\theta} \geq 0$ out of the total N_s samples.

Algorithm 9 presents the steps to calculate δ using the approach proposed in this theorem.

8.4.3 Obtaining Safe and Performant Value Function and Policy from $\hat{V}_\theta$

Using the δ-level estimate from Algorithm (9), we can finally obtain the safe and performant value function, $V_\theta(t, x)$, by solving the following epigraph optimization problem:

$$\begin{aligned} V_\theta(t, x) = \min_{z \in \mathbb{R}^+} \; & z \\ \text{s.t. } & \hat{V}_\theta(t, x, z) \leq \delta. \end{aligned} \tag{8.21}$$

Algorithm 10 Performance Quantification using Conformal Prediction

Require: Inputs: Safe set $\mathcal{S}^*$, calibration size N_p, confidence parameter β_p, target violation level ϵ_p, learned value V_θ, induced-policy value V_{π_θ}, normalizer $C_{\max}$
Ensure: Output: Calibrated normalized performance-error bound ψ

$D \leftarrow$ Sample N_p IID states from$\{x : x \in \mathcal{S}^*\}$
for $i = 0, 1, \ldots, N_p - 1$ **do**
    $P_i \leftarrow p_i(0, D)$
end for
$P \leftarrow P$ sorted in decreasing order
$\alpha_p \leftarrow \frac{1}{N_p+1}$, $\psi_0 \leftarrow P_0$, $\epsilon_0 \leftarrow$ (8.23)
for $i = 0, 1, \ldots, N_p - 1$ **do**
    while $\epsilon_i \leq \epsilon_p$ **do**
        $\alpha_p \leftarrow \frac{i+1}{N_p+1}$, $\psi_i \leftarrow P_i$, $\epsilon_i \leftarrow$ (8.23)
    end while
end for
return $\psi \leftarrow \psi_i$

Note that $V_\theta(t, x)$ is trivially ∞ for the states where $\hat{V}_\theta(t, x, z) > \delta$, since such states are unsafe and hence do not satisfy the safety constraint.

In practice, we solve this optimization problem by using a binary search approach on z. The resulting optimal state-feedback control policy, $\pi_\theta : \mathcal{X} \times [t, T) \to \mathcal{U}$, satisfying Objective (8.1), is given by:

$$\pi_\theta(t, x) = \arg\min_u \langle \nabla \hat{V}_\theta(t, \hat{x}^*), \hat{f}(\hat{x}^*, u) \rangle, \tag{8.22}$$

where $\hat{x}^*$ is the augmented state associated with the optimal z^* obtained by solving (8.21), i.e., $\hat{x}^* = [x, z^*]^T$. Intuitively, we can expect π_θ to learn behaviors that best tradeoff the safety and performance of the system.

8.4.4 Performance Quantification

In general, the learning inaccuracies in the auxiliary value function $\hat{V}_\theta$, may lead to errors in the value function V_θ. These errors, in turn, can lead to performance degradation under policy π_θ.

To quantify this degradation, we propose a conformal prediction-based performance quantification method that provides a probabilistic upper bound on the error between the value function and the value obtained from the induced policy. The following theorem formalizes our approach:

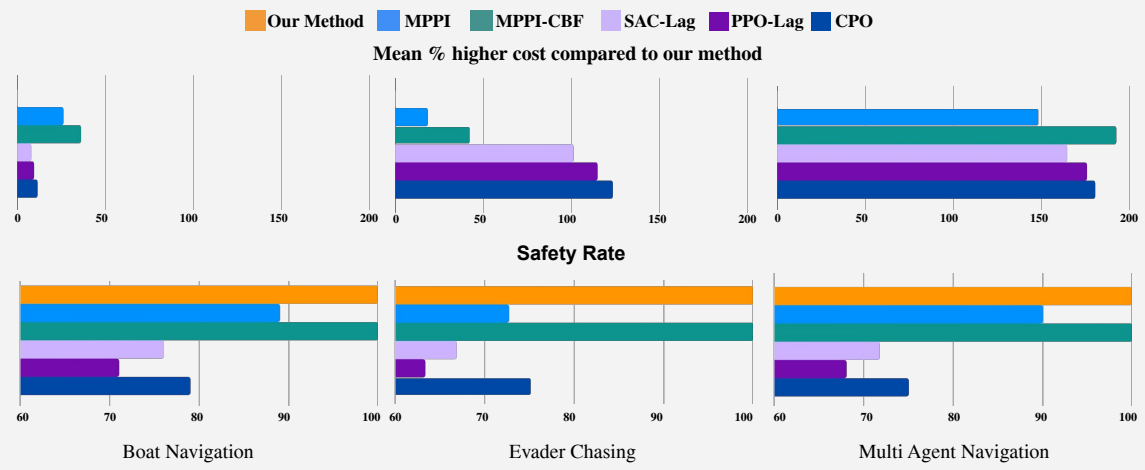


Figure 8.2: This figure presents a comparative study between all the methods based on our evaluation metrics. The top plot illustrates the **mean percentage increase in cumulative cost** relative to our method for each baseline, demonstrating that our approach consistently incurs lower costs, with the gap widening as system complexity grows. The bottom plot depicts the **safety rates**, showing that our method maintains a 100% safety rate, while baselines that encourage safety rather than enforcing it (like MPPI and C-SAC) achieve lower rates. MPPI-CBF also attains 100% safety but at the expense of performance. Overall, our method uniquely **balances both safety and performance**, whereas the baselines compromise on at least one aspect.

Theorem 8.4.2 (Performance Quantification Using Conformal Prediction)**.** *Suppose $\mathcal{S}^*$ denotes the safe states satisfying $V_\theta(0,x) < \infty$ (or equivalently $\hat{V}_\theta(0,\hat{x}^*) < \delta$) and $(0,x_i)_{i=1,\dots,N_p}$ are N_p i.i.d. samples from $\mathcal{S}^*$. For a user-specified level α_p, let ψ be the $\frac{\lceil (N_p+1)(1-\alpha_p)\rceil}{N_p}$th quantile of the scores $(p_i := \frac{|V_\theta(0,x_i)-V_{\pi_\theta}(0,x_i)|}{C_{max}})_{i=1,\dots,N_p}$ on the N_p state samples. Select a violation parameter $\epsilon_p \in (0,1)$ and a confidence parameter $\beta_p \in (0,1)$ such that:*

$$\sum_{i=0}^{l-1} \binom{N_p}{i} \epsilon_p^i (1-\epsilon_p)^{N_p-i} \leq \beta_p \tag{8.23}$$

where, $l = \lfloor (N_p+1)\alpha_p \rfloor$. Then, the following holds, with probability $1-\beta_p$:

$$\mathbb{P}_{x\in\mathcal{S}^*}\left(\frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}} \leq \psi\right) \geq 1-\epsilon_p. \tag{8.24}$$

where C_{max} is a normalizing factor and denotes the maximum possible cost that could be incurred for any $x \in \mathcal{S}^*$.

The proof is available in Appendix E.1.2. Note that C_{max} can be easily calculated by calculating the upper bound of the cost function $C(t,x(t),\mathbf{u})\forall x \in \mathcal{S}^*$.

Intuitively, the performance of the resultant policy is the best when the ψ value

approaches 0, while the worst performance occurs at $\psi = 1$. Algorithm 10 presents the steps to calculate ψ using the approach proposed in this theorem.

8.4.5 Numerical Simulations

The objective of this paper is to demonstrate the co-optimization of performance and safety. To achieve this, we evaluate the proposed method and compare them with baselines using three metrics: (1) **Cumulative Cost:** This metric represents the total cost $\int_0^T l(x(s))ds + \phi(x(T))$, accumulated by a policy over the safe trajectories. (2) **Safety Rate:** This metric is defined as the percentage of trajectories that remain safe, i.e., never enter the failure region $\mathcal{F}$ at any point in time. (3) **Computation Time:** This metric compares the offline and online computation times of our method and the baselines.

Baselines: We consider two categories of baselines: the first set of methods aims to enhance the system performance (i.e., minimize the cumulative cost) while encouraging safety, encompassing methods such as Lagrangian-based CRL algorithms like SAC-Lagrangian (SAC-Lag), PPO-Lagrangian (PPO-Lag) [130, 78] and Model Predictive Path Integral (MPPI) [185] algorithms. The second category prioritizes safety, potentially at the cost of performance. This includes Constrained Policy Optimization (CPO) [3] and safety filtering techniques such as Control Barrier Function (CBF)-based quadratic programs (QP) [8] that modify a nominal, potentially unsafe controller to satisfy the safety constraint.

8.4.6 Efficient and Safe Boat Navigation

In our first experiment, we consider a 2D autonomous boat navigation problem, where a boat with coordinates (x_b, y_b) navigates a river with state-dependent drift to reach an island. The boat must avoid two circular boulders (obstacles) of different radii, which corresponds to the safety constraint in the system (see Fig. 8.3). The cost function penalizes the distance to the goal. The system state, x, evolves according to the dynamics:

$$x = [x_b, y_b], \quad \dot{x} = [u_1 + 2 - 0.5y_b^2, u_2] \tag{8.25}$$

where $[u_1, u_2]$ are the bounded control inputs in the x_b and y_b directions, constrained by the control space $\mathcal{U} = \{[u_1, u_2] \in \mathbb{R}^2 \mid ||[u_1, u_2]|| \leq 1\}$. The term $2 - 0.5y_b^2$ introduces a state-dependent drift, complicating the control task as the actions must counteract the drift while ensuring safety, which is challenging under bounded control

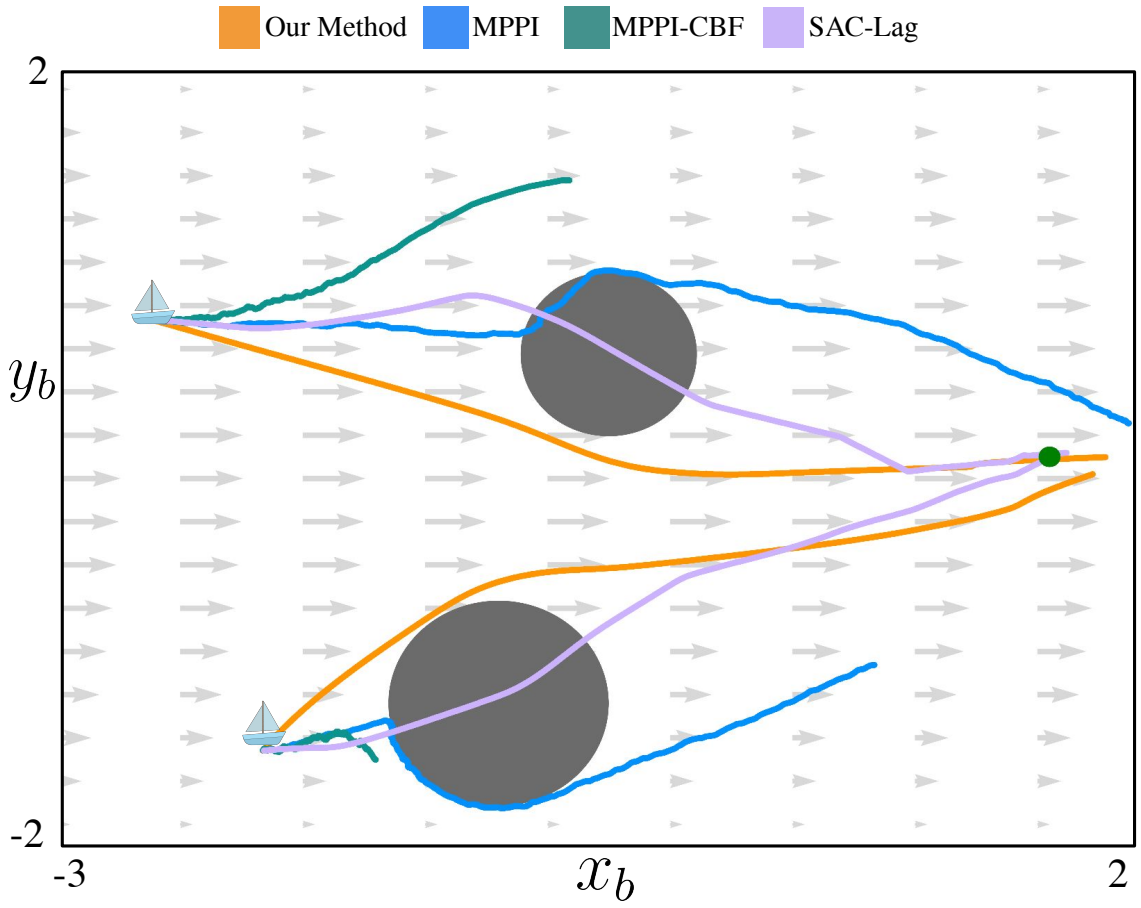


Figure 8.3: Trajectories from two distinct initial states are shown, with dark grey circles representing obstacles and the green dot indicating the goal at $[1.5, 0]^T$. Notably, our method is the **only one** that **successfully approaches the goal** while **adhering to safety constraints**.

inputs. The rest of the details about the experiment setup can be found in the Appendix E.2.1.

Safety Guarantees and Performance Quantification: We use $N_s = 300K$ and $N_p = 300K$ samples for thorough verification, ensuring dense state space sampling. For this experiment, we set $\epsilon_s = 0.001$ and $\beta_s = 10^{-10}$, resulting in a δ-level of 0. This implies that, with $1 - 10^{-10}$ confidence, any state with $\hat{V}_\theta(t, x, z) \leq 0$, is safe with at least 99.9% probability. For performance quantification, we set $\epsilon_p = 0.01$ and $\beta_p = 10^{-10}$, leading to a ψ-level of 0.136. This ensures, with $1-10^{-10}$ confidence, that any state in $\mathcal{S}^*$ has a normalized error between the predicted value and the policy value of less than 0.136 with 99% probability. Low δ and ψ values with high confidence indicate that the learned policy closely approximates the optimal policy and successfully co-optimizes safety and performance.

Baselines: This being a 2-dimensional system, we compare our method with the ground truth value function computed by solving the HJB-PDE numerically using

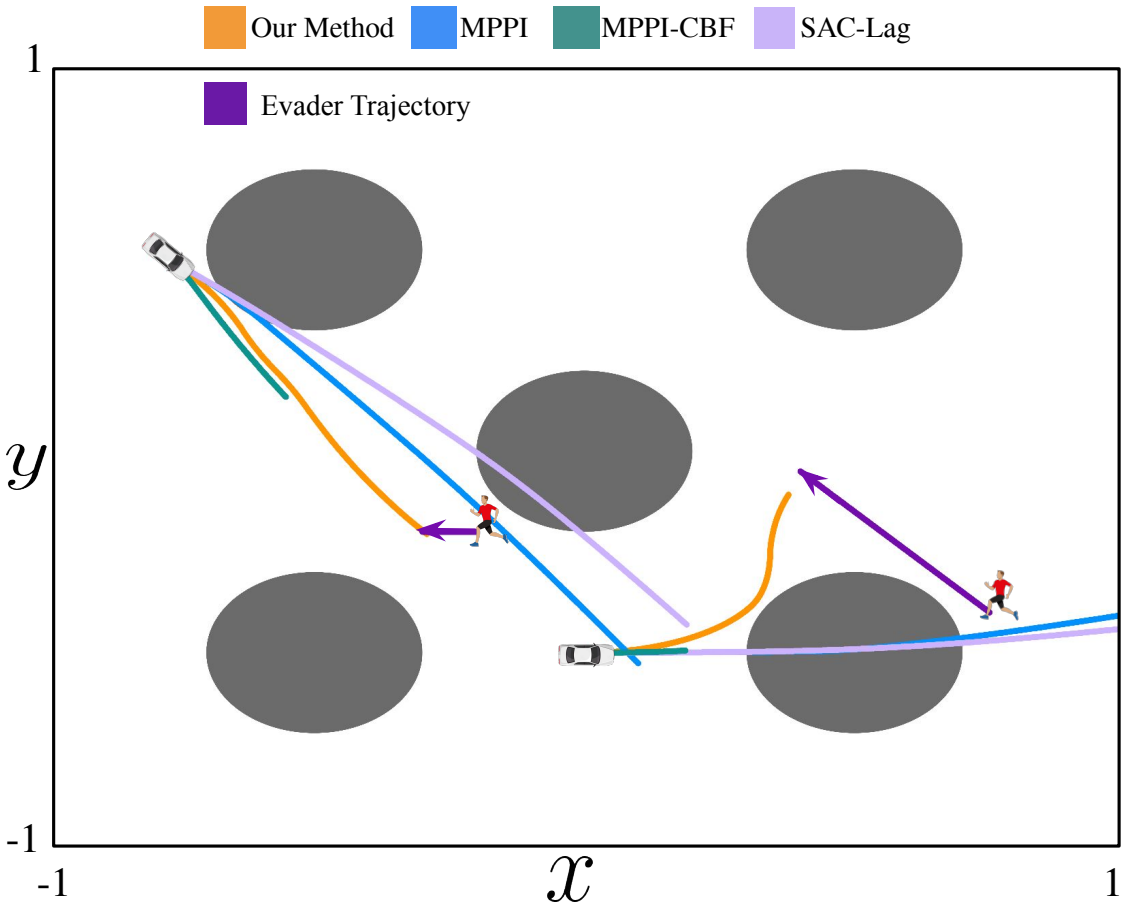


Figure 8.4: Trajectories from two distinct initial states are depicted, with dark grey circles representing obstacles and purple trajectories indicating the evader's path, with arrows showing its direction of motion. Our method successfully **tracks the evader** while **avoiding collisions**, whereas all other methods either fail to maintain safety, struggle to track the evader or both

the Level Set Toolbox [107] (results in Appendix E.2.1). Additional baselines include: (1) MPPI, a sample-based path-planning algorithm with safety as soft constraints, (2) MPPI-NCBF, where safety is enforced using a Neural CBF-based QP with MPPI as the nominal controller [46, 165], and (3) Constrained RL methods like SAC-Lag, PPO-Lag, and CPO.

Comparative Analysis: Figure 8.3 shows that our method effectively reaches the goal while avoiding obstacles, even when starting close to them. In contrast, MPPI and CRL-based policies fail to maintain safety, while MPPI-NCBF ensures safety but performs poorly (leading to very slow trajectories). Figure 8.2 highlights that our method outperforms all others. SAC-Lag attains a mean cost that is 7.5% higher than ours, while exhibiting the lowest safety rate at 76%. The remaining CRL methods display comparable trends, highlighting their inability to jointly optimize for safety and performance. MPPI, with a more competitive safety rate of 89%, performs poorly with a 32.67% higher mean cost. MPPI-NCBF achieves 100% safety

but performs significantly worse, with a 50.72% higher mean cost. Additionally, CBF-based controllers sometimes violate control bounds, limiting their applicability. This demonstrates that our method balances safety and performance, unlike others that compromise on one aspect. Moreover, the 100% safety rate of our method aligns closely with at least 99.9% safety level that we expect using our proposed verification strategy, providing empirical validation of the safety assurances.

8.4.7 Pursuer Vehicle tracking a moving Evader

In our second experiment, we consider an acceleration-driven pursuer vehicle, tracking a moving evader while avoiding five circular obstacles (see Fig. 8.4). This experiment involves an 8-dimensional system, with the state x defined as $x = [x_p, y_p, v, \Theta, x_e, y_e, v_{xe}, v_{ye}]^T$, where x_p, y_p, v, Θ represent the coordinates, linear velocity, and orientation of the pursuer vehicle, respectively, and x_e, y_e, v_{xe}, v_{ye} represent the coordinates and linear velocities of the evader vehicle. The pursuer vehicle is controlled by linear acceleration (u_1) and angular velocity (u_2). The control space is $\mathcal{U} = \{[u_1, u_2] \in [-2, 2]^2\}$. The complexity of this system stems from the dynamic nature of the goal, along with the challenge of ensuring safety in a cluttered environment, which in itself is a difficult safety problem. More details about the experiment setup are in Appendix E.2.2.

Safety Guarantees and Performance Quantification: Similar to the previous experiment, we set $N_s = N_p = 300k$. We choose $\epsilon_s = 0.01$ and $\beta_s = 10^{-10}$, yielding a δ-level of -0.04 and a safety level of 99% on the auxiliary value function. For performance, we set $\epsilon_p = 0.01$ and $\beta_p = 10^{-10}$, leading to a ψ-level of 0.137. These values indicate the learned policy maintains high safety with low-performance degradation in this cluttered environment.

Baselines: As in the previous experiment, we employ MPPI and CRL methods (SAC-Lag, PPO-Lag, and CPO). For safety filtering, we utilize a QP based on the collision cone CBF (C3BF) [154], chosen for its effectiveness in managing acceleration-driven systems.

Comparative Analysis: Figure 8.4 shows that our method effectively tracks the moving evader while avoiding obstacles, even when starting close to them. In contrast, other methods have limitations: MPPI and CRL methods attempt to follow the evader but fail to maintain their pace, violating safety constraints, while MPPI-C3BF sacrifices performance to maintain safety. Figure 8.2 highlights our method's superior performance in balancing safety and performance. MPPI achieves the best performance among the baselines but with an 18% higher mean cost and only a 72% safety rate. MPPI-NCBF ensures 100% safety but has a 42% higher mean cost. SAC-Lag underperforms both in safety (66% safety rate) and performance (101% higher

mean cost). A similar trend is evident across all other CRL methods, indicating their difficulty in co-optimizing safety and performance in high-dimensional, complex systems.

Receding Horizon Control: An interesting application of the synthesized policy is its deployment in a receding horizon fashion over a time horizon longer than that used for training the value function, as illustrated in Fig. 8.5. The results indicate that the learned policy successfully maintains safety while effectively tracking the evader over a 6-second horizon, despite being trained over a horizon of 1 second. This suggests that the proposed approach can be extended to effectively co-optimize safety and performance for long-horizon tasks by solving the SC-OCP over a shorter time horizon. Consequently, this framework offers a practical solution for real-world autonomous systems that require long-horizon safety and performance guarantees.

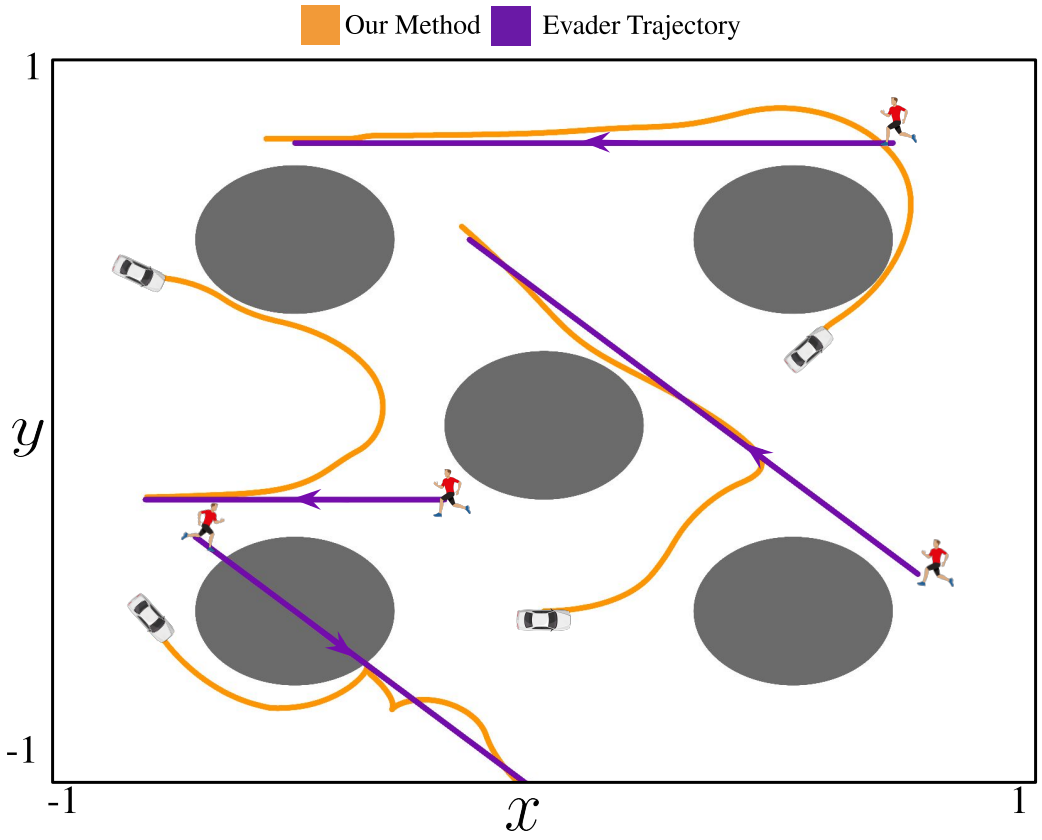


Figure 8.5: Trajectories of the **receding horizon policy** for the pursuer tracking an evader over a 6-second horizon, while the value function is trained over a 1-second horizon. The results demonstrate that the pursuer **successfully tracks** the evader while ensuring safety, even when initialized near the obstacle. This highlights the effectiveness of the proposed approach in **jointly optimizing safety and performance for long-horizon tasks.**

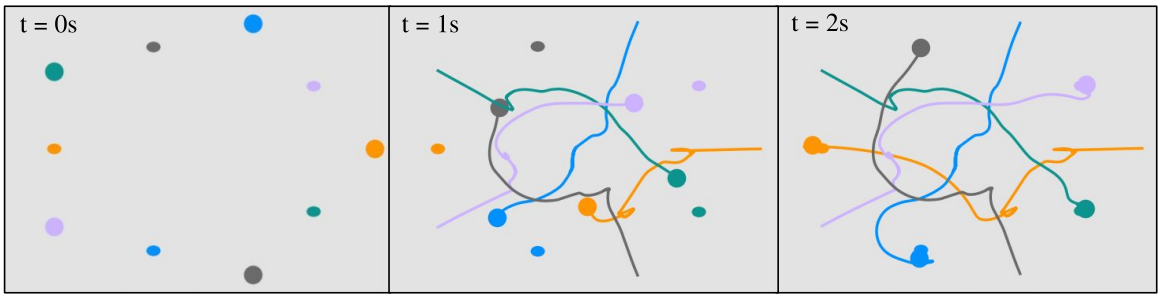


Figure 8.6: Snapshots of multi-agent navigation trajectories at different times using the proposed method. Agents are represented as circles with radius R, indicating the minimum safe distance they must maintain from each other. Smaller dots mark their respective goals. The trajectories show that agents proactively **maintain long-horizon safety** by adjusting their paths to avoid close encounters, rather than enforcing safety reactively, which could lead to suboptimal behaviors. Finally, the agents **reach their respective goals within the specified time horizon**.

8.4.8 Multi-Agent Navigation

In our third experiment, we consider a multi-agent setting where each of the 5 agents, represented by $x_i = [x_{a_i}, y_{a_i}, x_{g_i}, y_{g_i}]$, tries to reach its goal while avoiding collisions with others. (x_{a_i}, y_{a_i}) denote the position of the ith agent, while (x_{g_i}, y_{g_i}) represent the goal locations for that agent. The system is 20-dimensional, with each agent controlled by its x and y velocities. The control space for each agent is $\mathcal{U}_i = \{[v_{x_i}, v_{y_i}] \mid ||[v_{x_i}, v_{y_i}]|| \leq 1\}$. The complexity of this system stems from the interactions and potential conflicts between agents as they attempt to reach their goals while avoiding collisions. The rest of the details about the experiment setup can be found in Appendix E.2.3.

Safety Guarantees and Performance Quantification: We set $N_s = N_p = 300k$, $\epsilon_s = 0.001$, and $\beta_s = 10^{-10}$, resulting in a δ-level of -0.09 with safety assurance of 99.9% for the auxiliary value function. For performance quantification, we set $\epsilon_p = 0.01$ and $\beta_p = 10^{-10}$, leading to a ψ-level of 0.068. It is evident that the δ and ψ values remain very low with high confidence, highlighting the effectiveness of our method in co-optimizing safety and performance for high-dimensional, multi-agent systems.

Baselines: Similar to previous experiments, we have used MPPI, SAC-Lag, PPO-Lag, CPO, and MPPI-NCBF as our baselines for this experiment too.

Comparative Analysis: Figure 8.6 shows that our method ensures long-horizon safety while enabling all agents to reach their goals without collisions. In contrast, the baseline methods either exhibit overly conservative behavior or fail to maintain safety, leading to collisions, as detailed in Appendix E.2.3. Figure 8.2 demonstrates the superior performance of our approach, with MPPI, MPPI-NCBF, and SAC-Lag

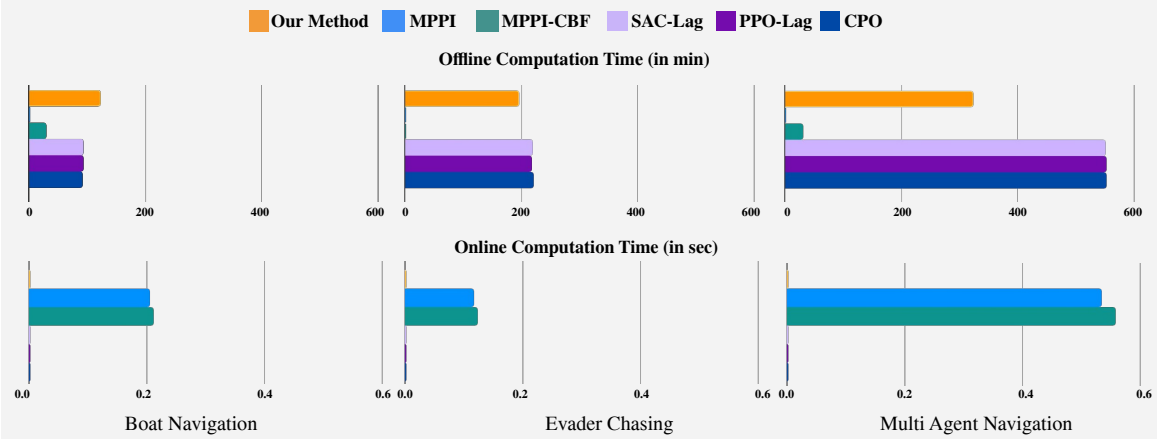


Figure 8.7: This figure presents a comparative analysis of all methods based on online and offline computation time evaluated on the same computing machine. The top plot illustrates the **offline computation time** for our method and the baselines. Since our method and SAC-Lag involve training value functions, they incur higher offline computation costs, whereas MPPI-based methods require no offline training. The bottom plot depicts the **online computation time**, demonstrating that our method and SAC-Lag have minimal online computation requirements, whereas MPPI-based methods exhibit significantly higher online computational costs.

showing mean percentage cost increases of 148%, 192%, and 164%, respectively. Although MPPI and MPPI-NCBF achieve competitive safety rates of 90% and 100%, their significant performance degradation highlights their inability to balance safety and performance in complex systems. MPPI's subpar performance stems from its reliance on locally optimal solutions in a finite data regime, leading to several deadlocks along the way and overall suboptimal trajectories over a long horizon. Furthermore, CRL methods struggle with both safety and performance, further demonstrating their limitations in handling increasing system complexity and dimensionality. These results confirm our method's ability to co-optimize safety and performance in high-dimensional systems, demonstrating its scalability. Additionally, the safety guarantees hold in the test samples, validating the scalability of our safety verification framework for multi-agent systems.

8.4.9 Computation time Analysis

Figure 8.7 presents a comparative analysis of the offline and online computation times for our method against the baselines. While traditional grid-based methods suffer from an exponentially scaling computational complexity (and are completely intractable for the 8D Evader Chasing and 20D Multi-Agent case studies), the proposed method scales much better with the system dimensionality. For example, the computation time increases only minimally from the 2D system to the 8D system, thanks to neural network parallelization. Similarly, the computation time increases sublinearly from

8D to 20D system. This scalability is a key advantage of the proposed approach. We finally note that while offline training requires time, our method achieves real-time inference speeds, with optimal policy computed in just **2ms** across all systems, making the approach highly suitable for real robotic systems.

Relation to MPC-style solutions. Our MPPI baselines are themselves receding-horizon (sampling-based MPC) optimizers, and Figure 8.7 quantifies that comparison directly: they attain comparable safety when given sufficient sampling budget but incur markedly higher *online* cost, since the optimization is re-solved at every time step. A gradient-based nonlinear MPC solver with hard state constraints could in principle match or exceed the achieved task cost on these low-dimensional examples; however, it would require online solve times, solver tuning, and recursive-feasibility machinery at every step, and would not share the dimension-scaling behavior demonstrated above. The proposed approach instead amortizes all optimization offline into a neural value function whose inference cost is essentially independent of system dimension, which is the operative trade-off for the high-dimensional case studies where grid- or optimization-based methods become intractable.

8.5 Safe and Optimal Control with Imperfect Information and Disturbances

While the preceding framework offers a principled means to co-optimize safety and performance in deterministic settings, real-world autonomous systems seldom operate without uncertainty. Such uncertainties may arise from imperfect state information, external disturbances, or other unmodeled effects, and their presence substantially complicates the co-optimization process by necessitating explicit reasoning about uncertainty. To address this challenge, we extend our analysis to the RSC-OCP introduced in Problem 8.1, which incorporates disturbances directly into the problem structure. This transition naturally guides us toward the primary objective of this work, as articulated in Objective 8.1. A tractable solution to the RSC-OCP equips our methodology with robustness to uncertainty, thereby enhancing its suitability for real-world deployment.

To solve the RSC-OCP in Equation (8.2), our goal is to compute the optimal robust value function V_r, which minimizes the cost while ensuring system safety. The theoretical background required for this formulation, together with its epigraph reformulation, is provided in Section 8.3. Building on this foundation, we now present a systematic procedure for computing V_r.

In the subsections that follow, we proceed in a structured manner. We first construct the robust auxiliary value function $\hat{V}_r$ using a physics-informed machine learning framework. We then apply a disturbance-aware conformal prediction procedure to certify safety and mitigate approximation errors in the learned auxiliary value function $\hat{V}_r$. The resulting safety-refined approximation is subsequently mapped to the final value function V_r via the epigraph formulation in (8.3). Finally, we assess the performance of the resulting value function using a modified conformal prediction–based evaluation method that accounts for disturbances.

8.5.1 Training the Robust Auxiliary Value Function ($\hat{V}_r$)

The robust auxiliary value function, $\hat{V}_r$, is learned using the same curriculum training scheme as described in the previous section. The DNN architectures and training details are the same as in the previous section. The proposed learning framework utilizes a loss function that enforces two primary objectives: (i) compliance with the PDE in (8.10), using the PDE residual error given by:

$$\begin{aligned}\mathcal{L}_{pde}\left(t_k, \hat{x}_k|\theta\right) = \| \min \Big\{ &-\partial_t \hat{V}_\theta\left(t_k, \hat{x}_k\right) - H(t_k, \hat{x}_k), \\ &\hat{V}_\theta\left(t_k, \hat{x}_k\right) - g\left(x_k\right)\Big\} \|,\end{aligned} \tag{8.26}$$

where $H(t, \hat{x}) = \min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \langle \nabla \hat{V}_\theta(t, \hat{x}), \hat{f}(\hat{x}, u, d)\rangle$ and (ii) satisfaction of the boundary condition in (8.11), using boundary condition loss, given by:

$$\begin{aligned}\mathcal{L}_{bc}\left(t_k, \hat{x}_k|\theta\right) = \|\max\left(\phi(x_k) - z_k, g(x_k)\right) - \\ \hat{V}_\theta\left(t_k, \hat{x}_k\right)\Big\| \mathbb{1}\left(t_k = T\right).\end{aligned} \tag{8.27}$$

These terms are balanced by a trade-off parameter λ, leading to the overall loss function:

$$\mathcal{L}\left(t_k, \hat{x}_k|\theta\right) = \mathcal{L}_{pde}\left(t_k, \hat{x}_k|\theta\right) + \lambda \mathcal{L}_{bc}\left(t_k, \hat{x}_k|\theta\right) \tag{8.28}$$

Furthermore, we use the adaptive loss re-balancing scheme proposed in [181] to reduce the impact of λ on the learned value function. Minimizing the overall loss function provides a self-supervised learning mechanism to approximate the auxiliary value function.

Algorithm 11 Safety Verification using Conformal Prediction

Require: Inputs: Robust safe-state domain $\mathcal{S}$, calibration size N_s, confidence parameter β_s, target violation level ϵ_s, learned value $\hat{V}_\theta$, worst-case induced value $\hat{V}_{\hat{\pi}_\theta}$, number of candidate levels M

Ensure: Output: Disturbance-aware safety-correction level δ

$D_0 \leftarrow$ Sample N_s IID states from $\mathcal{S}_{\delta=0}$
$\delta_0 \leftarrow \min_{\hat{x}_j \in D_0} \left\{ \hat{V}_\theta(0, \hat{x}_j) : \hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_j) \geq 0 \right\}$
$\epsilon_0 \leftarrow$ (8.33) (using $\alpha_{\delta=0}$)
$\Delta \leftarrow$ Ordered list of M uniform samples from $[\delta_0, 0]$
for $i = 0, 1, \ldots, M-1$ **do**
    while $\epsilon_i \leq \epsilon_s$ **do**
        $\delta_i \leftarrow \Delta_i$
        Update α_{δ_i} from δ_i
        $\epsilon_i \leftarrow$ (8.33) (using α_{δ_i})
    end while
end for
return $\delta \leftarrow \delta_i$

8.5.2 Safety Verification

The learned auxiliary value function, $\hat{V}_\theta$, induces a policy, $\hat{\pi}_\theta$, that minimizes the Hamiltonian term $H(t, \hat{x})$ in the HJB-PDE. The policy is given by:

$$\hat{\pi}_\theta(t, \hat{x}) = \arg\min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \langle \nabla \hat{V}_\theta(t, \hat{x}), \hat{f}(\hat{x}, u, d) \rangle. \tag{8.29}$$

$$\hat{d}_\theta(t, \hat{x}) = \arg\max_{d \in \mathcal{D}} \min_{u \in \mathcal{U}} \langle \nabla \hat{V}_\theta(t, \hat{x}), \hat{f}(\hat{x}, u, d) \rangle. \tag{8.30}$$

The rollout cost corresponding to this policy is defined as:

$$\begin{aligned} \hat{V}_{\hat{\pi}_\theta}(t, \hat{x}) = \max \big\{ & C(t, x(t), \mathbf{u}, \mathbf{d}) - z, \\ & \max_{s \in [t,T]} g(x(s)) \big\} \Big|_{\substack{\mathbf{u}=\hat{\pi}_\theta, \\ \mathbf{d}=\hat{d}_\theta}} \end{aligned} \tag{8.31}$$

Ideally, the rollout cost from a given state under $\hat{\pi}_\theta$ should match the value of the auxiliary value function at that state. However, due to learning inaccuracies, discrepancies can arise. This becomes critical when a state, $\hat{x}_i$, is deemed safe by the auxiliary value function ($\hat{V}_\theta(t, \hat{x}) \leq 0$) but is unsafe under the induced policy ($\hat{V}_{\hat{\pi}_\theta}(t, \hat{x}) > 0$). To address this, we introduce a uniform value function correction margin, δ, which guarantees that the sub-δ level set of the auxiliary value function remains safe under the induced policy. Mathematically, the optimal δ (δ^*) can be expressed as:

$$\delta^* := \min_{\hat{x} \in \mathcal{X}} \{ \hat{V}_\theta(0, \hat{x}) : \hat{V}_{\hat{\pi}_\theta}(0, \hat{x}) \geq 0 \} \tag{8.32}$$

Intuitively, δ^* identifies the tightest level of the value function that separates safe states under $\hat{\pi}_\theta$ from unsafe ones. Hence, any initial state within the sub-δ^* level set is guaranteed to be safe under the induced policy, $\hat{\pi}^*_\theta$. However, calculating δ^* exactly requires infinitely many state-space points. To overcome this, we adopt a conformal-prediction-based approach to approximate δ^* using a finite number of samples, providing a probabilistic safety guarantee. The following theorem formalizes our approach:

Theorem 8.5.1 (Safety Verification Under Bounded Disturbances)**.** *Consider a system subject to bounded disturbances $d \in \mathcal{D}$, with the induced rollout cost $\hat{V}_{\hat{\pi}_\theta}$ evaluated under both the learned control policy $\hat{\pi}_\theta$ and the learned worst-case disturbance policy $\hat{d}_\theta$, as defined in (8.31). Let $\mathcal{S}_\delta$ be the set of states satisfying $\hat{V}_\theta(0, \hat{x}) \leq \delta$, and let $(0, \hat{x}_i)_{i=1,\dots,N_s}$ be N_s i.i.d. samples from $\mathcal{S}_\delta$. Define α_δ as the safety error rate among these N_s samples for a given δ level. Select a safety violation parameter $\epsilon_s \in (0,1)$ and a confidence parameter $\beta_s \in (0,1)$ such that:*

$$\sum_{i=0}^{l-1} \binom{N_s}{i} \epsilon_s^i (1-\epsilon_s)^{N_s - i} \leq \beta_s, \tag{8.33}$$

where $l = \lfloor (N_s+1)\alpha_\delta \rfloor$. Then, with the probability of at least $1-\beta_s$, the following holds:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s. \tag{8.34}$$

This result generalizes Theorem 8.4.1 to the robust setting; when $\mathcal{D} = \{0\}$, the disturbance vanishes and the guarantee reduces to Theorem 8.4.1.

The proof is available in Appendix E.1.3. The safety error rate α_δ is defined as the fraction of samples satisfying $\hat{V}_\theta \leq \delta$ and $\hat{V}_{\hat{\pi}_\theta} \geq 0$ out of the total N_s samples.

Algorithm 11 presents the steps to calculate δ using the approach proposed in this theorem.

8.5.3 Obtaining Safe and Performant Value Function and Policy from $\hat{V}_\theta$

Using the δ-level estimate from Algorithm (11), we can finally obtain the safe and performant value function, $V_\theta(t,x)$, by solving the following epigraph optimization problem:

$$\begin{aligned} V_\theta(t,x) = \min_{z \in \mathbb{R}^+} \; & z \\ \text{s.t. } & \hat{V}_\theta(t,x,z) \leq \delta. \end{aligned} \tag{8.35}$$

Algorithm 12 Performance Quantification using Conformal Prediction

Require: Inputs: Robust safe set $\mathcal{S}^*$, calibration size N_p, confidence parameter β_p, target violation level ϵ_p, learned value V_θ, worst-case induced value V_{π_θ}, normalizer $C_{\max}$

Ensure: Output: Disturbance-aware normalized performance-error bound ψ

$D \leftarrow$ Sample N_p IID states from $\{x : x \in \mathcal{S}^*\}$
for $i = 0, 1, \ldots, N_p - 1$ **do**
    $P_i \leftarrow p_i(0, D)$
end for
$P \leftarrow P$ sorted in decreasing order
$\alpha_p \leftarrow \frac{1}{N_p+1}$, $\psi_0 \leftarrow P_0$, $\epsilon_0 \leftarrow$ (8.37)
for $i = 0, 1, \ldots, N_p - 1$ **do**
    while $\epsilon_i \leq \epsilon_p$ **do**
        $\alpha_p \leftarrow \frac{i+1}{N_p+1}$, $\psi_i \leftarrow P_i$, $\epsilon_i \leftarrow$ (8.37)
    end while
end for
return $\psi \leftarrow \psi_i$

Note that $V_\theta(t, x)$ is trivially ∞ for the states where $\hat{V}_\theta(t, x, z) > \delta$, since such states are unsafe and hence do not satisfy the safety constraint.

In practice, we solve this optimization problem by using a binary search approach on z. The resulting optimal state-feedback control policy, $\pi_\theta : \mathcal{X} \times [t, T) \to \mathcal{U}$, satisfying Objective (8.1), is given by:

$$\pi_\theta(t, x) = \arg\min_u \langle \nabla \hat{V}_\theta(t, \hat{x}^*), \hat{f}(\hat{x}^*, u) \rangle, \tag{8.36}$$

where $\hat{x}^*$ is the augmented state associated with the optimal z^* obtained by solving (8.35), i.e., $\hat{x}^* = [x, z^*]^T$. Intuitively, we can expect π_θ to learn behaviors that best tradeoff the safety and performance of the system.

8.5.4 Performance Quantification

In general, the learning inaccuracies in the auxiliary value function $\hat{V}_\theta$, may lead to errors in the value function V_θ. These errors, in turn, can lead to performance degradation under policy π_θ.

To quantify this degradation, we propose a conformal prediction-based performance quantification method that provides a probabilistic upper bound on the error between the value function and the value obtained from the induced policy. The following theorem formalizes our approach:

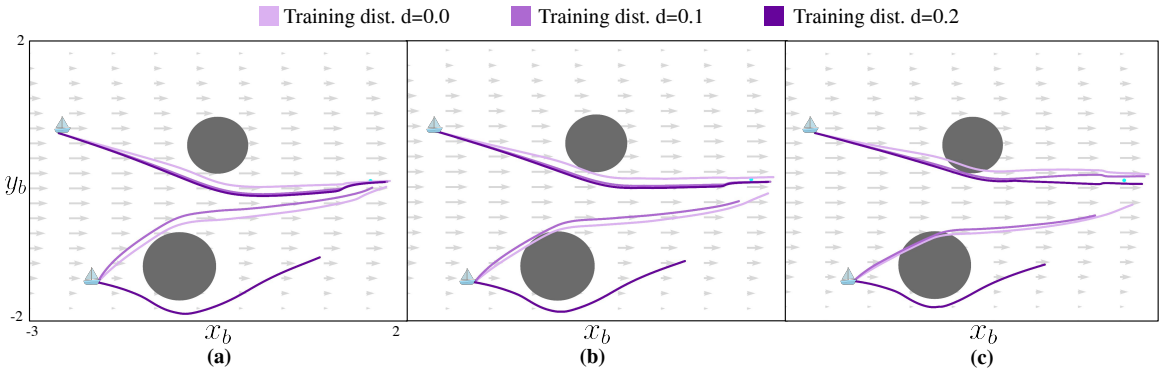


Figure 8.8: Trajectories from two different initial conditions are shown for value functions trained under varying disturbance levels and evaluated in simulation with different disturbance magnitudes. (a) Evaluation disturbance $d = 0.0$. (b) Evaluation disturbance $d = 0.1$. (c) Evaluation disturbance $d = 0.2$. The results indicate that policies trained with disturbance levels greater than or equal to those encountered during evaluation exhibit improved safety, highlighting the importance of learning robust value functions for maintaining safety under uncertainty. Additionally, policies trained with disturbance levels exceeding those present in the environment exhibit increased conservatism, although the performance degradation remains modest. These results demonstrate that the proposed framework can synthesize safe policies for uncertain environments without substantial performance loss.

Theorem 8.5.2 (Performance Quantification Under Bounded Disturbances)**.** *Consider a system subject to bounded disturbances $d \in \mathcal{D}$, where the induced value function $V_{\pi_\theta}(0, x)$ is obtained by rolling out the learned policy π_θ (from (8.36)) against the learned worst-case disturbance $\hat{d}_\theta$ (from (8.30)). Suppose $\mathcal{S}^*$ denotes the safe states satisfying $V_\theta(0, x) < \infty$ (or equivalently $\hat{V}_\theta(0, \hat{x}^*) < \delta$) and $(0, x_i)_{i=1,\ldots,N_p}$ are N_p i.i.d. samples from $\mathcal{S}^*$. For a user-specified level α_p, let ψ be the $\frac{\lceil (N_p+1)(1-\alpha_p) \rceil}{N_p}$th quantile of the scores $(p_i := \frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}})_{i=1,\ldots,N_p}$ on the N_p state samples. Select a violation parameter $\epsilon_p \in (0, 1)$ and a confidence parameter $\beta_p \in (0, 1)$ such that:*

$$\sum_{i=0}^{l-1} \binom{N_p}{i} \epsilon_p^i (1 - \epsilon_p)^{N_p - i} \leq \beta_p \tag{8.37}$$

where, $l = \lfloor (N_p + 1)\alpha_p \rfloor$. Then, the following holds, with probability $1 - \beta_p$:

$$\mathbb{P}_{x \in \mathcal{S}^*} \left(\frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}} \leq \psi \right) \geq 1 - \epsilon_p. \tag{8.38}$$

where C_{max} is a normalizing factor and denotes the maximum possible cost that could be incurred for any $x \in \mathcal{S}^*$. This result generalizes Theorem 8.4.2 to the robust setting; when $\mathcal{D} = \{0\}$, it reduces to Theorem 8.4.2.

The proof is available in Appendix E.1.4. Note that C_{max} can be easily calculated by calculating the upper bound of the cost function $C(t, x(t), \mathbf{u}) \forall x \in \mathcal{S}^*$.

Intuitively, the performance of the resultant policy is the best when the ψ value approaches 0, while the worst performance occurs at $\psi = 1$. Algorithm 12 presents the steps to calculate ψ using the approach proposed in this theorem.

8.5.5 Numerical Simulations

We evaluate the proposed framework under disturbances and partial model uncertainty using representative numerical simulation studies.

8.5.6 Efficient and Safe Boat Navigation

We revisit the boat navigation task and extend it to explicitly account for model uncertainty. The system dynamics are given by

$$x = \begin{bmatrix} x_b \\ y_b \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} u_1 + 2 - 0.5y_b^2 + d_1 \\ u_2 + d_2 \end{bmatrix} \tag{8.39}$$

where $[u_1, u_2]$ denote the control inputs and $[d_1, d_2]$ represent exogenous disturbances acting along the x_b and y_b directions, respectively. The control inputs are constrained to the admissible set $\mathcal{U} = \{[u_1, u_2] \in \mathbb{R}^2 \mid ||[u_1, u_2]|| \leq 1\}$, while the disturbances are assumed to be bounded such that $\mathcal{D} = \{[d_1, d_2] \in \mathbb{R}^2 \mid |d_i| \leq 0.1, 0.2\}, i \in \{1, 2\}$.

To study the impact of disturbance modeling on safety and performance, we train three separate value functions: one assuming nominal dynamics without disturbances, and two corresponding to different disturbance magnitudes. Each value function is then evaluated across all three disturbance settings, allowing us to assess both safety rates and task performance under matched and mismatched disturbance assumptions.

Tables 8.1 summarize the safety and performance of the learned value functions under varying levels of disturbance. When trained under nominal dynamics ($d = 0$), the value function achieves perfect safety in the disturbance-free setting but exhibits a progressive degradation in safety as the evaluation disturbance increases, indicating limited robustness to unmodeled uncertainty.

In contrast, value functions trained with nonzero disturbance bounds demonstrate improved generalization. Training with $d = 0.1$ preserves safety under matched and

Training Disturbance	δ-Level	Evaluation Disturbance					
		$d = 0$		$d = 0.1$		$d = 0.2$	
		Safety Rate ↑	Mean Cost ↓	Safety Rate ↑	Mean Cost ↓	Safety Rate ↑	Mean Cost ↓
$d = 0.0$	0.000	100 %	2.88	89 %	2.99	80 %	3.15
$d = 0.1$	−0.018	100 %	2.97	100 %	3.10	88 %	3.19
$d = 0.2$	−0.028	100 %	2.96	100 %	3.14	100 %	3.28

Table 8.1: Safety rate (%) and mean task cost for the boat navigation case study under varying disturbance levels. A notable trend is that the safety rate remains at 100% when the evaluation disturbance is equal to or smaller than the disturbance used during training, but decreases when the evaluation disturbance exceeds the training disturbance. Additionally, increasing the training disturbance does not lead to a substantial increase in the mean task cost, suggesting that the proposed robust training framework improves safety without significantly degrading task performance.

lower disturbance levels and partially mitigates safety degradation under higher disturbances. The value function trained with the largest disturbance bound ($d = 0.2$) maintains 100% safety across all evaluation settings, suggesting that incorporating worst-case disturbance assumptions during training leads to more robust safety preservation.

This increased robustness is accompanied by a modest increase in mean task cost, reflecting more conservative behavior. However, the cost increase remains gradual across disturbance levels, indicating that robustness is achieved without a substantial loss in efficiency. This trend is clearly illustrated in Figure 8.8. Overall, these results highlight a clear trade-off between conservatism and performance, and demonstrate that disturbance-aware value synthesis improves safety generalization under uncertainty.

Safety Guarantees and Performance Quantification: To evaluate the validity of the safety guarantees in the robust setting, we conduct an ablation study comparing safety rates with and without adjusting the value function using the δ-levels obtained from the CP-based safety verification procedure. Consistent with the previous disturbance-free boat navigation study, we set $N_s = N_p = 300k$, $\epsilon_s = 0.001$, and $\beta_s = 10^{-10}$. The verification procedure is performed under the same disturbance bounds used during training. As shown in Table 8.2, adjusting the value function using the computed δ-level improves the observed safety rates relative to the case without safety verification. While the guarantees in the robust setting apply only to the policy-induced value of states, unlike the stronger guarantees on the true state values obtained in the non-robust setting, the results demonstrate that the verification step still provides meaningful improvements in safety rates.

We also quantify performance on the robust value functions trained under the three disturbance scenarios, obtaining ψ-levels of 0.137, 0.139, and 0.143, respec-

Training Dist.	w/o CP	w/ CP
$d = 0.0$	100%	100%
$d = 0.1$	98%	100%
$d = 0.2$	95%	100%

Table 8.2: Safety rate (%) with and without conformal prediction (CP) correction. Each row reports the safety rate evaluated at its respective training disturbance level.

tively. These results indicate that disturbances do not substantially increase training difficulty and that the proposed approach learns value functions that maintain strong performance under disturbances.

8.5.7 Pursuer Vehicle tracking an Uncertain moving Evader

Next, we consider a tracking problem in which a pursuer aims to track an uncertain moving evader. The pursuer dynamics are given by

$$\dot{x}_p = v\cos\Theta, \quad \dot{y}_p = v\sin\Theta, \quad \dot{v} = u_1, \quad \dot{\Theta} = u_2, \tag{8.40}$$

where (x_p, y_p) denote the pursuer position, v its speed, Θ its heading, and u_1, u_2 are bounded control inputs. The evader dynamics are modeled as

$$\begin{aligned} \dot{x}_e = v_{xe} + d_1, \quad \dot{y}_e = v_{ye} + d_2, \\ \dot{v}_{xe} = d_{v1}, \quad \dot{v}_{ye} = d_{v2}, \end{aligned} \tag{8.41}$$

where (x_e, y_e) denote the evader position, (v_{xe}, v_{ye}) its velocity components, and d_i and d_{vi} represent bounded disturbances capturing uncertainty in the evader model. The disturbances are assumed to satisfy $|d_1|, |d_2| \leq 0.18, |d_{v1}|, |d_{v2}| \leq 0.08$, corresponding to approximately 20% of the respective state bounds.

We train two value functions: one assuming nominal evader dynamics without disturbances, and one incorporating the disturbance bounds above. For evaluation, we simulate environments under three disturbance settings: zero disturbance ($d = 0$), the nominal disturbance bounds (denoted dist), and twice the nominal disturbance magnitude (2 dist, corresponding to 40% of the state bounds). Each value function is evaluated over trajectories initialized from 1000 randomly sampled initial conditions.

Figure 8.9 illustrates that the robust policy adapts more effectively to uncertainty in the evader's behavior compared to the non-robust policy. This improved adaptability results in more consistent tracking performance, thereby demonstrating the effectiveness of the robust policy in uncertain environments. Table 8.3 summarizes

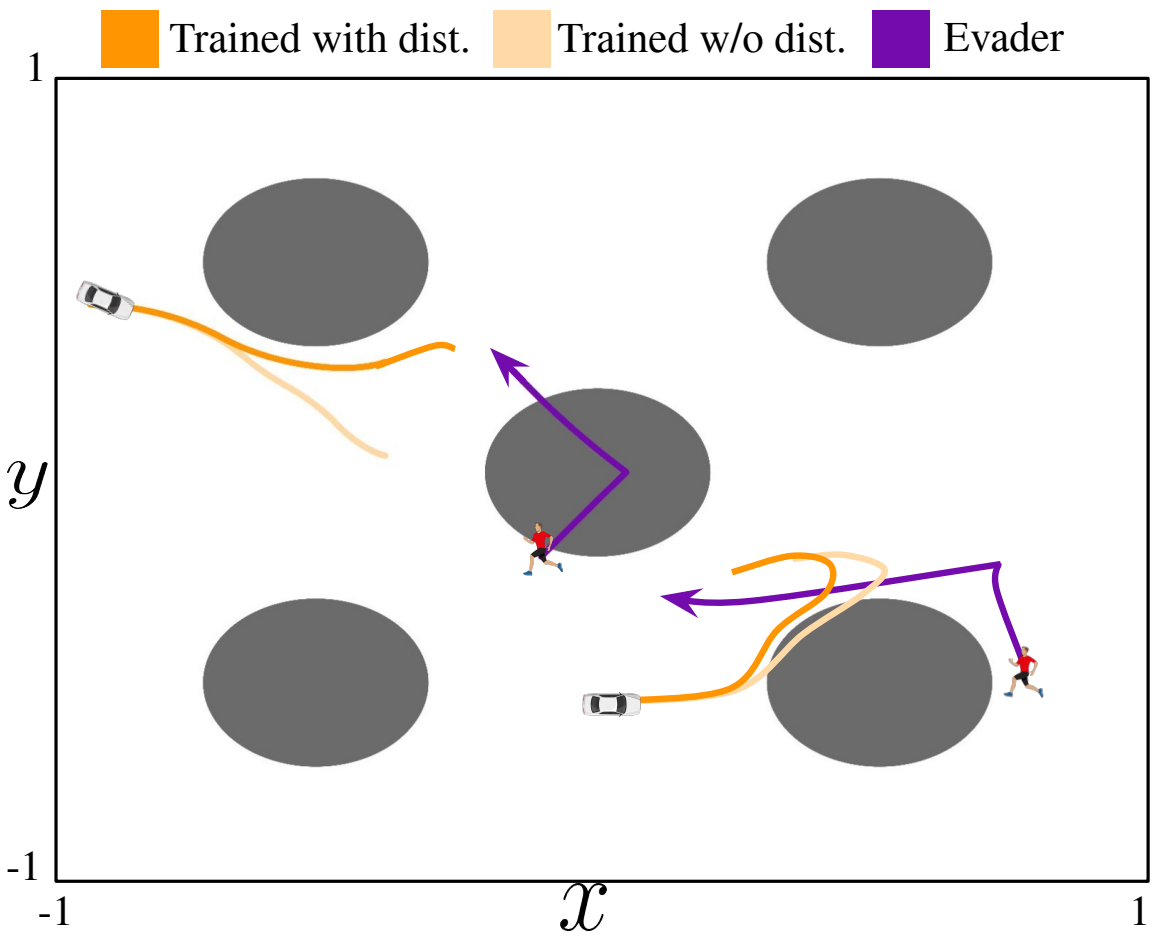


Figure 8.9: Trajectories from two distinct initial conditions are shown for value functions trained with and without disturbance in the evader model. The policy derived from the robust value function effectively adapts to the evader's erratic behavior arising from model uncertainty and successfully maintains tracking. In contrast, the non-robust policy exhibits limited adaptation, resulting in less effective tracking and consequently higher tracking costs.

the safety and performance of the learned value functions under increasing levels of evader model uncertainty. When trained without disturbances, the value function achieves perfect safety under nominal conditions but exhibits a progressive degradation in safety as the disturbance magnitude increases, indicating limited robustness to modeling errors in the evader dynamics. In contrast, incorporating disturbance bounds during training substantially improves safety generalization, maintaining near-perfect safety under matched disturbance conditions and significantly higher safety under amplified disturbances. This robustness is achieved with a modest increase in mean task cost, reflecting more conservative tracking behavior. Overall, these results demonstrate that explicitly modeling evader uncertainty during value synthesis improves safety preservation under both expected and unforeseen disturbance levels.

Safety Guarantees and Performance Quantification: Similar to the robust

Training Setting	δ-Level	Evaluation Disturbance					
		$d = 0$		$d = \text{dist}$		$d = 2 \times \text{dist}$	
		Safety Rate ↑	Mean Cost ↓	Safety Rate ↑	Mean Cost ↓	Safety Rate ↑	Mean Cost ↓
No disturbance	-0.04	100.0 %	1.65	88.6 %	1.70	83.8 %	1.78
With disturbance	-0.06	100.0 %	1.81	99.2 %	1.81	92.4 %	1.86

Table 8.3: Safety rate (%) and mean task cost for the evader tracking case study with and without evader model uncertainty. A notable trend is that the safety rate remains high when the evaluation disturbance is equal to or smaller than the training disturbance, but decreases when the evaluation disturbance exceeds it. Additionally, increasing the training disturbance does not lead to a substantial increase in the mean task cost, suggesting that the proposed robust training framework, when trained with high disturbance values, improves safety without significantly degrading task performance.

boat navigation case study, we perform an ablation study to examine the effectiveness of the proposed safety verification scheme. As in the disturbance-free evader tracking experiment, we set $N_s = N_p = 300k$, $\epsilon_s = 0.01$, and $\beta_s = 10^{-10}$.The results in Table 8.4 show a trend consistent with the boat navigation case study: applying safety verification leads to higher safety rates. This observation further demonstrates the effectiveness of the proposed CP-based verification scheme, even in the presence of disturbances.

Training Dist.	w/o CP	w/ CP
No disturbance	97.6%	100.0%
With disturbance	94.8%	99.2%

Table 8.4: Safety rate (%) with and without conformal prediction (CP) correction for the pursuer-evader tracking case study. Each row reports the safety rate evaluated at its respective training disturbance level.

In addition, for performance evaluation, we obtain ψ-levels of 0.136 and 0.141 for the disturbance and disturbance-free settings, respectively. These results indicate that the proposed robust training strategy, when coupled with the CP-based safety verification scheme, enables the synthesis of value functions that achieve both strong safety rates and high performance.

8.6 Hardware Experiments

To validate the proposed robust framework's applicability for real-world deployment, we apply it to a real hardware testbed. Specifically, we deploy the pursuer-evader tracking task, where a pursuer must track a moving evader while avoiding circular

obstacles, mirroring the simulation setup in Sections 8.4 and 8.5. The experimental setup is shown in Fig. 8.10.

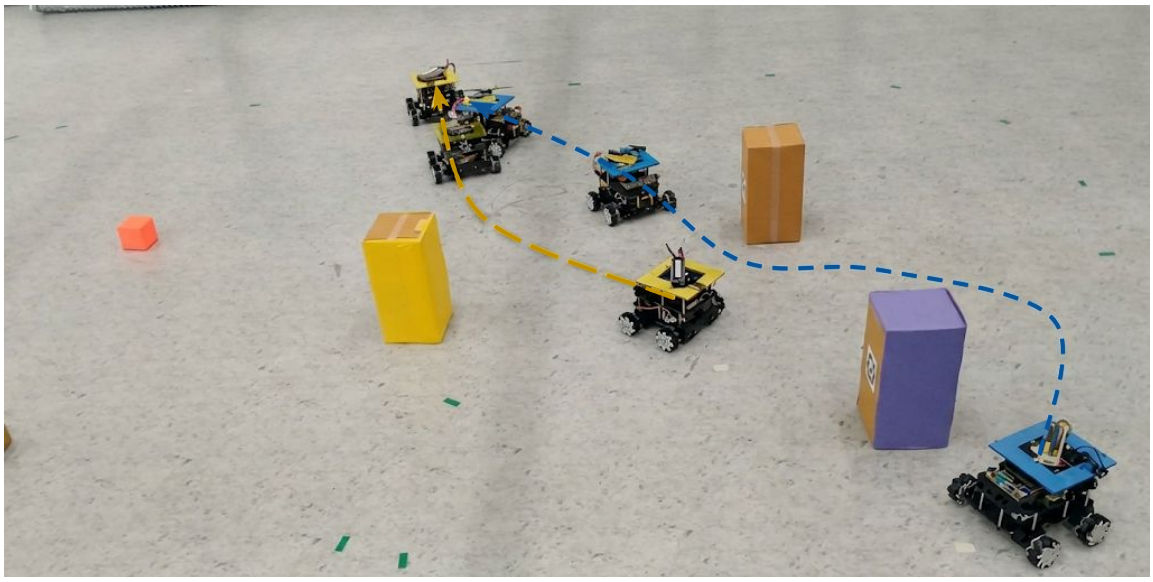


Figure 8.10: Hardware experiment setup for the pursuer-evader tracking task. Two customized TurtleBot3 Burger platforms fitted with mecanum wheels are used: the blue-topped robot (pursuer, unicycle dynamics) tracks the yellow-topped robot (evader, point-mass dynamics) while avoiding box-shaped obstacles. The dashed lines indicate representative trajectories of the pursuer (blue) and evader (yellow). Global localization is provided by the PhaseSpace™ motion capture system, and policy inference is performed on an offboard PC that sends control commands to both robots in real time.

We use two customized TurtleBot3 Burger platforms ($138 \times 178 \times 192$ mm, 1 kg each), fitted with mecanum wheels to enable omnidirectional motion (Fig. 8.10). Each robot is identified by a colored top: the blue robot serves as the pursuer, and the yellow robot serves as the evader. The pursuer operates under unicycle dynamics with state $[x_p, y_p, v, \Theta]^T$, controlled by linear acceleration (u_1) and angular velocity (u_2), consistent with the model described in Section 8.4. The maximum linear speed is 0.22 m/s and the maximum angular velocity is 2.84 rad/s. The evader operates under point-mass dynamics with state $[x_e, y_e, v_{xe}, v_{ye}]^T$, controlled by velocity inputs (v_x, v_y) with a maximum speed of 0.22 m/s; the mecanum wheels allow it to realize point-mass dynamics directly. The global positions and orientations of both robots are measured using a PhaseSpace™ motion capture system at a tracking frequency of 960 Hz, with two LED markers placed on the front and rear of each robot for state estimation. Velocities are obtained via finite differencing of the position measurements, filtered using a low-pass filter to reduce noise.

The robust auxiliary value function $\hat{V}_\theta$ is trained offline using the evader disturbance bounds described in Section 8.5, together with additional disturbance bounds on the pursuer states, $|d_{x_p}|, |d_{y_p}|, |d_v| \leq 0.2$ and $|d_\Theta| \leq \pi/5$. These limits correspond

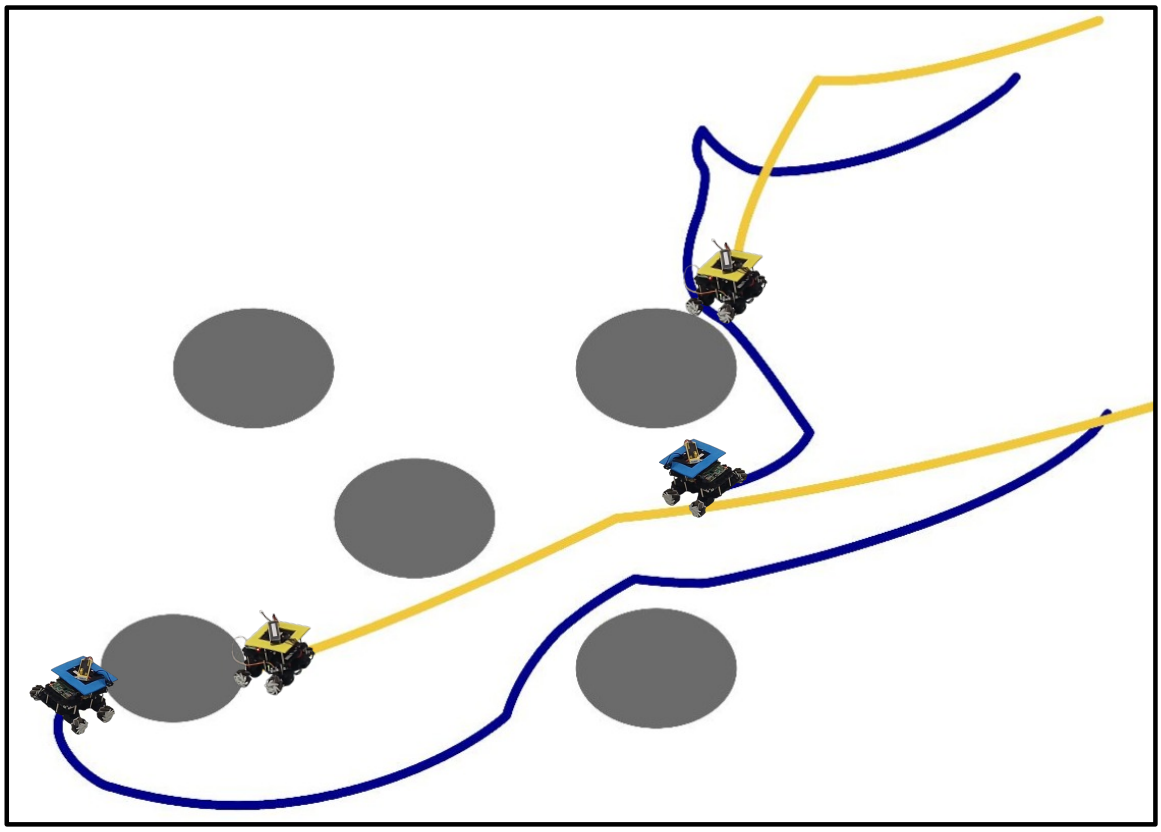


Figure 8.11: This figure presents two representative trajectories obtained from the hardware experiments. The pursuer adapts to the evader's erratic motion and maintains successful tracking, demonstrating the robustness of the proposed approach to disturbances encountered during sim-to-real transfer and its potential for real-world deployment.

to approximately 20% of the respective state bounds and are introduced to account for potential modeling inaccuracies in the pursuer dynamics. The resulting value function is subsequently used for online policy inference, with the policy executed in a receding-horizon manner, as described in Section 8.4.7, enabling effective tracking of the evader over time horizons longer than those used during training. At each control step, the current state estimate is transmitted to an offboard PC, which evaluates the learned policy π_θ via (8.36). The resulting control commands are sent to the robots at a frequency of 100 Hz, with policy inference taking approximately 2 ms per step, well within the 10 ms control period. The experiments are conducted in a cluttered environment with obstacles of average size 0.25 m. The evader follows preplanned trajectories that pass near obstacles, creating challenging tracking scenarios that stress both the safety and performance of the deployed policy. We evaluate the policy from 10 distinct initial conditions to assess repeatability.

Across 10 experimental trials, the pursuer consistently tracks the evader while satisfying all safety constraints, resulting in a 100% safety rate and a mean tracking error of 3.67 m. Figure 8.11 presents two representative trials. As illustrated, the

deployed robust policy produces smooth, non-conservative trajectories that closely follow the evader, even in the presence of nearby obstacles, which is consistent with the behavior observed in simulation. We attribute the effective sim-to-real transfer to two factors: (i) the use of a disturbance-aware value function, which provides inherent robustness to modeling inaccuracies, sensor noise, and actuation delays not explicitly modeled during training; and (ii) the high control frequency (100 Hz), which mitigates discretization effects. These results demonstrate that the proposed framework can be deployed on physical robotic systems in real time, and that the robust formulation introduced in Section 8.5 is instrumental in bridging the sim-to-real gap for safety-critical autonomy.

8.7 Chapter Summary

This chapter developed a physics-informed machine learning framework for jointly co-optimizing safety and performance in autonomous systems. By reformulating the state-constrained optimal control problem via an epigraph transformation and solving the resulting HJB PDE with neural networks, we obtained auxiliary value functions from which both safe and near-optimal policies can be extracted. Conformal prediction provided formal safety certificates and performance bounds. The framework was demonstrated on systems ranging from 2D to 20D, achieving real-time inference with strong safety and optimality guarantees.

The multi-vehicle case study in this chapter already hinted at the potential for multi-agent applications. However, the centralized formulation used therein treats all agents as a single high-dimensional system, which does not scale to large numbers of agents. In the next chapter, we develop a *decentralized* extension of the physics-informed framework that enables each agent to solve a fixed-dimensional optimization problem while maintaining global safety and performance. The key innovation is a reachability-based neighbor-selection strategy that identifies the most safety-critical interactions without requiring full knowledge of the global state.

Chapter 9

Scaling Safety–Performance Co-optimization to Multi-Agent Systems

9.1 Related Works

Chapter 8 demonstrated that physics-informed learning can co-optimize safety and performance for single-agent systems up to moderate dimensionality. However, many real-world applications (warehouse logistics, autonomous intersection management, drone swarms) involve *multiple* interacting agents that must coordinate safely. Extending safe optimal control to multi-agent settings introduces fundamental scalability challenges: the joint state space grows linearly with the number of agents, making centralized approaches intractable.

Multi-Agent Safe Control. Classical approaches to multi-agent collision avoidance include velocity obstacles [17], ORCA [16], and potential-field methods [83]. CBF-based multi-agent safety has been explored through pairwise barrier constraints [179, 21], where each pair of agents contributes a constraint to the QP. While theoretically sound, pairwise CBF methods scale quadratically with the number of agents and require hand-designed barrier functions.

Multi-Agent Reinforcement Learning. MARL methods [102, 193] learn decentralized policies through interaction, but rarely provide formal safety guarantees. Constrained MARL approaches [62] incorporate safety through reward shaping or Lagrangian penalties, which remain soft constraints. Mean-field approximations [192]

improve scalability but sacrifice the ability to reason about specific pairwise interactions.

Scalable HJ Methods. Decomposition strategies for multi-agent HJ reachability [30] partition the joint system into subsystems, but the decomposition is typically fixed and may not capture the most safety-critical interactions. Learning-based HJ approaches [13] can handle moderate dimensions but have not been extended to the combinatorial structure of multi-agent problems.

Gap Addressed. This chapter introduces a decentralized physics-informed framework that maintains the formal safety–optimality structure of Chapter 8 while scaling to systems with many agents. The key insight is that each agent solves a *fixed-dimensional* optimization problem by attending only to its most safety-critical neighbors, selected via a principled reachability-based criterion.

9.2 Contributions of this Chapter

This chapter makes the following contributions:

1. We reformulate the multi-agent SC-OCP into a *decentralized* formulation with a *fixed observation size*, ensuring that each agent's optimization problem has constant dimensionality regardless of the total number of agents.
2. We introduce a *reachability-based neighbor selection* strategy that uses pairwise safety value functions to identify the most safety-critical interactions, prioritizing agents on a collision course over merely nearby ones.
3. We evaluate the framework from 3 to 16 agents in head-to-head comparisons with baselines—reporting safety rate, task cost, and the fraction of fully safe scenarios—and further evaluate the same decentralized policy, with unchanged network weights and fixed local observation size, in environments with up to 256 agents. These are empirical scalability results under the stated density, observation-radius, and neighbor-selection settings.
4. Ablation studies validate the importance of the decentralized formulation, the HJB-informed training, and the reachability-based neighbor selection.

The results presented in this chapter are based on [161]. Additional material is available on the project webpage[1]. The accompanying code repository[2] is currently private.

9.3 Problem Setup

Consider a *homogeneous* multi-agent system comprising N agents. The state and control input for each agent i are defined as $x_i \in \mathcal{X}_i \subseteq \mathbb{R}^D$ and $u_i \in \mathcal{U}_i \subseteq \mathbb{R}^M$, respectively. Each agent's motion is governed by nonlinear dynamics expressed as: $\dot{x}_i(t) = f_i(x_i(t), u_i(t))$, where $f_i : \mathbb{R}^D \times \mathbb{R}^M \to \mathbb{R}^D$ is a locally Lipschitz continuous function. Although our framework assumes f_i is known, we note that this function could also be learned from data if a model is not available.

Using this premise, we define the joint state and control input vectors for the entire system as $x := [x_1, \ldots, x_N] \in \mathcal{X} \subseteq \mathbb{R}^{N \times D}$ and $u := [u_1, \ldots, u_N] \in \mathcal{U} \subseteq \mathbb{R}^{N \times M}$. Consequently, the collective system dynamics are described by the concatenated function $f = [f_1, f_2, \ldots, f_N]$, yielding the global dynamics model: $\dot{x}(t) = f(x(t), u(t))$.

The primary safety objective is to ensure collision avoidance between all agents. We formalize this by defining a failure set $\mathcal{F} \subseteq \mathcal{X}$ of unsafe configurations:

$$\mathcal{F} = \{x \in \mathcal{X} : \min_{i \neq j} d_{ij}(x) \leq r\}. \tag{9.1}$$

Here, d_{ij} is the Euclidean distance between agents i and j, and r is a predefined minimum safe separation distance. To integrate this safety requirement into the optimal control framework, we define a constraint function $g(x)$ such that $\mathcal{F} = \{x \in \mathcal{X} \mid g(x) > 0\}$. The system's performance is quantified by a cost functional $C(t, x(t), u(\cdot))$ that accumulates a running cost and a terminal cost over a time horizon:

$$C(t, x(t), u(\cdot)) = \int_{s=t}^{T} l(x(s))ds + \phi(x(T)), \tag{9.2}$$

where $l : \mathcal{X} \to \mathbb{R}_{\geq 0}$ and $\phi : \mathcal{X} \to \mathbb{R}_{\geq 0}$ are non-negative, Lipschitz continuous functions. The control input is $u(\cdot) : [t, T) \to \mathcal{U}$. C could represent the time taken to reach the goal or fuel consumption, for instance. We also assume the cost functions are separable and identical across the homogeneous agents.

Our goal is to synthesize an optimal control policy $\pi^* : [t, T) \times \mathcal{X} \to \mathcal{U}$ that minimizes this cost while guaranteeing that the state trajectory never enters the

[1] https://tayalmanan28.github.io/mad-pinn/
[2] https://github.com/tayalmanan28/mad-pinn

failure set $\mathcal{F}$. This leads to the following **State-Constrained Optimal Control Problem (SC-OCP)** with the value function $V(t, x)$:

$$\begin{aligned} V(t, x(t)) &= \min_{u(\cdot)} \int_t^T l(x(s))ds + \phi(x(T)) \\ \text{s.t.} \quad \dot{x}(s) &= f(x(s), u(s)), \quad g(x(s)) \leq 0 \ \ \forall s \in [t, T] \end{aligned} \tag{9.3}$$

Thus, the policy, π^*, derived from the solution of this SC-OCP co-optimizes safety and performance.

9.3.1 Decentralized Multi-Agent SC-OCP

As a direct consequence of its formulation, the centralized SC-OCP in (9.3) suffers from the curse of dimensionality; its computational complexity grows intractably with the number of agents N [160]. To address this scalability challenge, we introduce a decentralized reformulation of the problem.

In the decentralized framework, agents are constrained by a limited observation field. Rather than observing the global state x, each agent i must rely on a local observation o_i. This observation is constructed from the agent's own state and the states of a fixed number n of neighbouring agents within its observation radius r_{obs}, where $n \leq N$. Formally, the observation is constructed by an operator $\mathcal{O}_i$ that selects and concatenates these relevant states, i.e., $o_i = \mathcal{O}_i(x) \subseteq \mathbb{R}^{D\times(n+1)}$, where: $o_i = [x_i, \{x_j\}_{j\in\mathcal{N}_i}]$ and $\mathcal{N}_i$ denotes the set of indices corresponding to the neighbors of agent i.

A key enabling factor for this approach is the homogeneity of the agent dynamics and objectives. This property ensures that the decentralized value function is identical for all agents, irrespective of their specific identity. We can therefore drop the agent subscript i and represent a generic local observation simply as o. This allows us to define a single, shared decentralized optimal control problem (**Decentralized SC-OCP**):

$$\begin{aligned} V_d(t, o(t)) &= \min_{\mathbf{u}_d} \int_t^T l_d(o(s))\, ds + \phi_d(o(T)) \\ \text{s.t. } \dot{o}(s) &= f_d(o(s), u_d(s)),\ g_d(o(s)) \leq 0 \ \ \forall s \in [t, T] \end{aligned} \tag{9.4}$$

Here, l_d and ϕ_d represent the running and terminal cost functions, respectively, aggregated over the subset of agents within o. The control sequence $\mathbf{u}_d$ is the joint input for these agents. The dynamics function f_d is derived by concatenating the individual agent dynamics, and the constraint function g_d enforces safety (e.g., collision avoidance) among the agents in the local observation.

This reformulation addresses the scalability issue by fixing the observation size to $n+1$, rendering the local optimization dimension independent of the total agent count N. Consequently, a single value function V_d and its associated policy, computed once, can be used across agents. It provides a computationally tractable decentralized approximation to the original large-scale SC-OCP in (9.3); global safety additionally depends on the observation radius and neighbor-selection protocol covering every safety-critical interaction.

9.3.2 Epigraph Reformulation

Directly solving the State-Constrained Optimal Control Problem (SC-OCP) in (9.4) is challenging due to the presence of hard state constraints. To circumvent this challenge, we adopt an epigraph reformulation [22] that transforms the original constrained problem into a more tractable, equivalent two-stage optimization formulation. The core of this reformulation is the introduction of a non-negative auxiliary variable $z \in \mathbb{R}^+$, representing a bound on the cost-to-go. The original SC-OCP is then equivalently expressed as:

$$V_d(t, o(t)) = \min_{z \in \mathbb{R}^+} z \quad \text{s.t.} \quad \hat{V}_d(t, o, z) \leq 0, \tag{9.5}$$

where $\hat{V}_d$ denotes a newly defined auxiliary value function. Following the framework of [4], this function is given by:

$$\hat{V}_d(t, o(t), z) = \min_{\mathbf{u}_d} \max \left\{ C(t, o(t), \mathbf{u}_d) - z,\ \max_{s \in [t,T]} g_d(o(s)) \right\}. \tag{9.6}$$

This formulation captures the problem's dual objectives: minimizing cost and ensuring safety. Importantly, the condition $\hat{V}_d(t, o, z) < 0$ guarantees that $g_d(o(s)) < 0$ for all $s \in [t, T]$, meaning the system's trajectory remains within the safe set.

The optimal solution z^* to (9.5) represents the *minimum admissible cost* achievable without violating the state constraints. This interpretation provides an intuitive safety-performance trade-off: a choice of $z > z^*$ results in an overly conservative policy, while $z < z^*$ prioritizes performance at the potential expense of safety.

To further enable the application of dynamic programming principles, we treat the auxiliary variable z as a state variable with the simple dynamics $\dot{z}(t) = -l(o(t))$. This signifies that the admissible cost bound z is depleted by the stage cost $l(o)$ along the system's trajectory. This yields the following augmented system dynamics:

$$\dot{\hat{o}} = \hat{f}_d(t, \hat{o}, u) := \begin{bmatrix} f_d(t, o, u) \\ -l(o) \end{bmatrix}, \tag{9.7}$$

where $\hat{o} := [o, z]^\top \in \mathcal{X} \times \mathbb{R}$ is the augmented state.

Under standard assumptions A1–A4 from [4], the auxiliary value function $\hat{V}_d(t, \hat{o})$ is the unique continuous viscosity solution to the Hamilton-Jacobi-Bellman (HJB) partial differential equation:

$$\min\left(-\partial_t \hat{V}_d - \min_{\mathbf{u}_d} \left\langle \nabla_{\hat{o}} \hat{V}_d,\ \hat{f}_d(\hat{o}, u) \right\rangle,\ \hat{V}_d - g_d(o)\right) = 0, \tag{9.8}$$

for all $t \in [0, T)$ and $\hat{o} \in \mathcal{X} \times \mathbb{R}$, with the terminal condition:

$$\hat{V}_d(T, \hat{o}) = \max\left(\phi_d(o) - z,\ g_d(o)\right). \tag{9.9}$$

This HJB characterization provides a formal basis for computing the value function and the resulting optimal safe control policy. For notational brevity, we have dropped the subscript d and will hereafter use $\hat{V}(t, \hat{o})$ to denote this decentralized auxiliary value function.

9.4 Methodology

The solution to the SC-OCP formulated in Equation (9.3) hinges on the computation of the optimal value function V_d that minimizes cost under safety constraints. Our approach to obtaining this function proceeds in two primary stages as illustrated in Figure 9.1: an offline learning phase and an online deployment phase. First, we learn the auxiliary value function $\hat{V}$ using a physics-informed machine learning framework. Then, V_d is obtained from $\hat{V}$ using (9.5). For online decentralized deployment, a safety-aware clustering strategy is employed to determine the appropriate neighbours for each agent. The control policy for each agent is then derived based on V_d. We now discuss each step in detail.

9.4.1 Training the Auxiliary Value Function ($\hat{V}$)

The auxiliary value function $\hat{V}$ is characterized by the HJB-PDE in (9.8) (Section 9.3.2). Traditional numerical methods for solving such PDEs rely on discretizing the state space over a grid [107, 135]. While accurate for low-dimensional systems, these methods are susceptible to the curse of dimensionality, as their computational cost scales exponentially with the number of states. To overcome this limitation, we use a physics-informed neural network (PINN) framework that uses PDE residuals to learn the value function and has shown promising results in solving high-dimensional HJB PDEs [13].

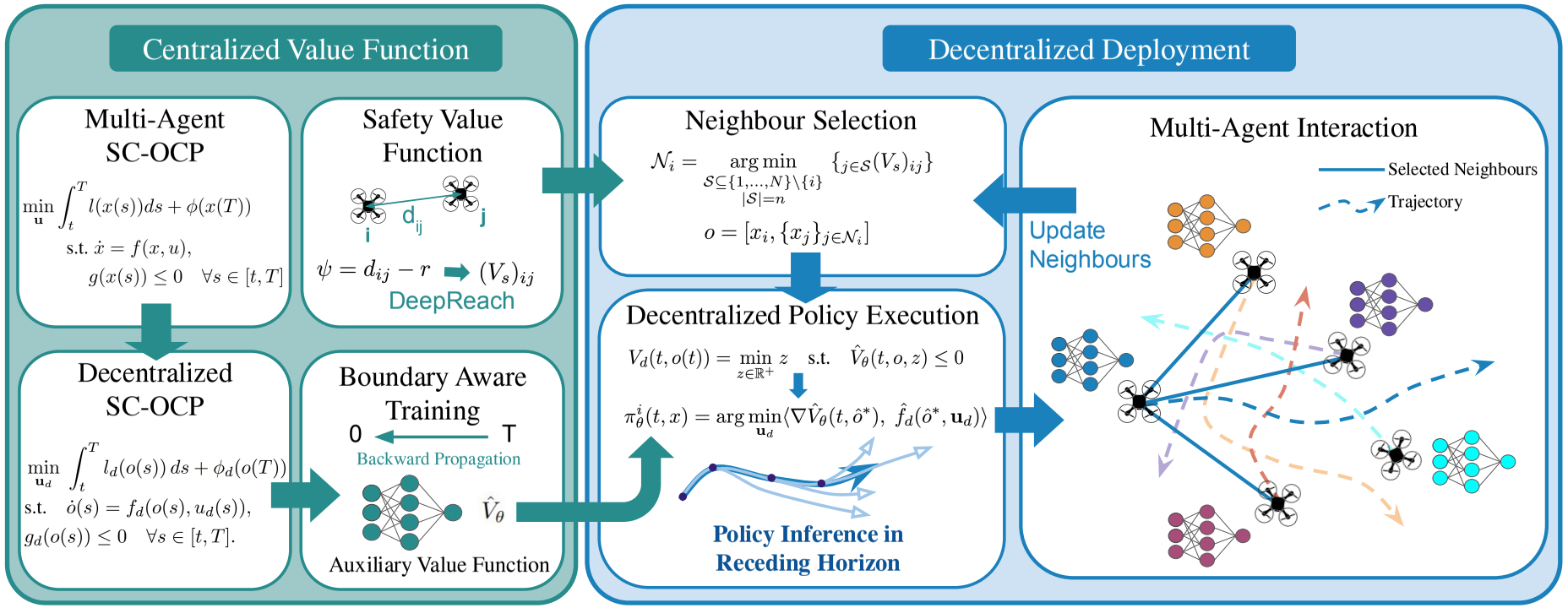


Figure 9.1: We propose **MAD-PINN** – a framework for safe and optimal multi-agent control. MAD-PINN is divided into two phases: (1) **Centralized training**, where we learn the auxiliary epigraph-based value function $\hat{V}_\theta$ using a boundary-aware PINN, and the pairwise safety value function V_s via DeepReach; (2) **Decentralized deployment**, where each agent selects its safety-critical neighbours using V_s and executes the policy in a receding-horizon manner using $\hat{V}_\theta$. This design enables tractable training, adaptive neighbour selection, and scalable execution in large multi-agent systems.

Auxiliary Value Function Parameterization: We approximate the auxiliary value function $\hat{V}(t, \hat{o})$ using a neural network, with parameters θ. A critical requirement is that the solution must satisfy the terminal boundary condition to adhere to the problem's safety constraints. To enforce this *exactly*, we structure our network output as:

$$\hat{V}_\theta(t, \hat{o}) = \max\left(\phi_d(o) - z,\ g_d(o)\right) + (T - t) \cdot R_\theta(t, \hat{o}),$$

where the first term encodes the terminal condition in (9.9), and the neural network, denoted $R_\theta(t, \hat{o})$, learns the residual evolution of the value function over time. This formulation, inspired by [139], guarantees that $\hat{V}_\theta(T, \hat{o}) = \max\left(\phi_d(o) - z,\ g_d(o)\right)$ for any state $\hat{o}$, irrespective of the network's output R_θ. There are two key advantages to the proposed structure of $\hat{V}_\theta$: (a) it eliminates the need to explicitly learn a complex boundary condition by hard-coding it into the network's forward pass, and (b) it reduces the learning problem to minimizing a single HJB-derived loss function (as we discuss next), thereby removing the necessity for a manually-tuned loss weighting scheme.

Loss Function and Training Scheme: The parameters θ of the network R_θ are learned by minimizing a loss function that penalizes the HJB PDE residual errors. Specifically, the loss function to learn the NN parameters is:

$$\begin{aligned}
&\mathcal{L}\left(t_k, \hat{o}_k | \theta\right) = \mathcal{L}_{pde}\left(t_k, \hat{o}_k | \theta\right) \\
&= \| \min\left\{-\partial_t \hat{V}_\theta\left(t_k, \hat{o}_k\right) - H(t_k, \hat{o}_k),\ \hat{V}_\theta\left(t_k, \hat{o}_k\right) - g_d\left(\hat{o}_k\right)\right\} \|, \\
&= \| \min\left\{(t_k - T)\partial_t R_\theta\left(\hat{o}_k, t_k\right) + R_\theta\left(\hat{o}_k, t_k\right) - H\left(\hat{o}_k, t_k\right),\right. \\
&\qquad \left. V_\theta\left(\hat{o}_k, t_k\right) - g_d\left(\hat{o}_k\right)\right\} \|,
\end{aligned} \tag{9.10}$$

where, $H(t, \hat{x}) = \min_{u \in \mathcal{U}} \langle \nabla V_\theta(\hat{o}_i, t), \hat{f}_d(\hat{o}_i, u) \rangle$. Typically, PINNs incorporate an additional loss term to enforce boundary conditions. In contrast, due to the structure of our formulation of $\hat{V}_\theta$, the boundary conditions are satisfied exactly. Consequently, the optimization reduces to minimizing a single loss term $\mathcal{L}_{pde}$, eliminating the need for auxiliary loss terms and the hyperparameters required for their weighting.

Curriculum Training: A key challenge in solving the HJB-PDE is its backward-in-time evolution; the solution at time t depends on the future time $t + \Delta t$. To manage this complexity during training, we employ a curriculum learning strategy similar to DeepReach [13]. Training begins by sampling time points near the terminal time T and progressively expanding the interval backward until it covers the entire horizon $[0, T]$, whereas the states are sampled uniformly across the state space at each training iteration. This allows the network to first learn the well-defined terminal condition accurately before learning to propagate the solution backward in time, governed by the PDE dynamics, yielding the auxiliary value function $\hat{V}_\theta$. We refer interested readers

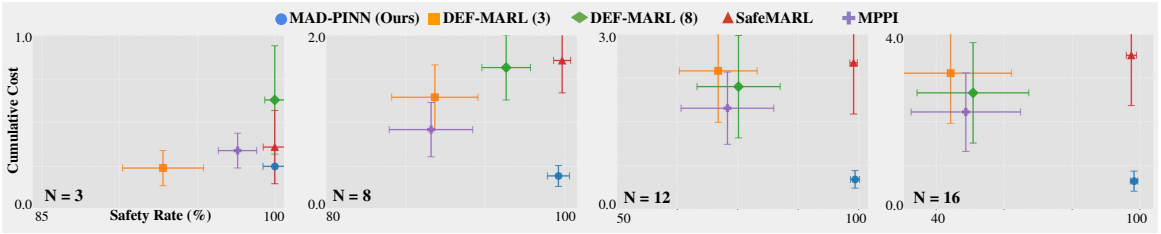


Figure 9.2: Comparison of cumulative costs and safety rates across testing environments with 3, 8, 12, and 16 agents. Our method consistently appears in the bottom-right region, indicating **superior safety-performance co-optimization relative to all baselines**. Its degradation in performance and safety with increasing agent count is minimal, **demonstrating better scalability compared to the baselines**.

to [13, 139] for more details on the curriculum training scheme.

9.4.2 Neighbour Selection Strategy

During online deployment, each agent must efficiently identify the subset of neighbours most relevant for maintaining safety. Rather than treating all nearby agents equally, we group agents into clusters defined by their potential for safety-critical interactions. The cluster size is chosen to match the neighbourhood dimension used when training the decentralized auxiliary value function, ensuring consistency between training and deployment.

To determine which neighbours pose the greatest risk, we employ a principled criterion derived from Hamilton–Jacobi (HJ) reachability analysis. HJ reachability provides a rigorous way to quantify the likelihood of safety conflicts, enabling each agent to prioritize interactions that most directly impact safe operation. Specifically, while $\hat{V}$ and V_d encode cost and safety constraints in the SC-OCP, we additionally compute a pairwise safety value function V_s using HJ reachability to guide neighbour selection.

HJ Reachability[108, 104] characterizes the set of states from which the system can be driven into a failure set. Let $\psi : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz-continuous function whose sub-zero level set $\Psi = \{x : \psi(x) \leq 0\}$ represents pairwise collision states between the two agents. The corresponding safety value function is given by:

$$V_s(x,t) = \sup_{u(\cdot)} \min_{\tau \in [t,T]} \psi(\xi^u_{x,t}(\tau)), \tag{9.11}$$

where $\xi^u_{x,t}(\cdot)$ is the system trajectory. Intuitively, $V_s(x,t)$ measures how close the two agents are to a collision, even under optimal control. The unsafe set is precisely the sub-zero level set of V_s. The value function satisfies the Hamilton-Jacobi-Bellman

Variational Inequality (HJB-VI):

$$\begin{aligned} \min\{\partial_t V_s(x,t) + H_s(x,t), \psi(x) - V_s(x,t)\} = 0, \\ V_s(x,T) = \psi(x), \\ H_s(x,t) = \max_{u \in \mathcal{U}} \langle \nabla V_s(x,t) \ , \ f_s(x,u) \rangle, \end{aligned} \tag{9.12}$$

where H_s is the Hamiltonian corresponding to the safety value function and f_s encodes the pairwise dynamics between agents.

Learning the Pairwise Safety Value Function (V_s): We approximate V_s using DeepReach [13, 139]. To characterize pairwise safety interactions, we consider each agent i in relation to another agent j. The target function is defined as $\psi = d_{ij} - r$, where d_{ij} denotes the distance between the two agents and r is the prescribed collision radius. By design, $\psi \leq 0$ corresponds to states in which agents i and j are in collision, thereby providing a natural safety signal. Training in this manner yields a value function that quantifies the relative safety risk posed by one agent to another.

Neighbour Selection: For agent i, the value $(V_s)_{ij}$ represents the degree of risk posed by agent j. Smaller values correspond to higher collision likelihoods. Specifically, if $(V_s)_{ij} < (V_s)_{ik}$, then agent j poses a higher safety risk to agent i compared to agent k, and thus should be prioritized in i's decision-making process. Hence, to select its n neighbours, agent i computes $(V_s)_{ij}$ for all agents j within its observation radius r_{obs}, ranks them, and selects the n agents with the lowest values. This process ensures that each agent focuses its decision-making on the most safety-critical interactions, while still keeping the neighbourhood size fixed for computational tractability.

9.4.3 Policy Synthesis

Once the neighbour set is determined, each agent synthesizes its policy by solving the optimization problem in (9.5). To enforce safety, we set $V_d(t,x) = +\infty$ whenever $\hat{V}_\theta(t,x,z) > 0$, since such states are unsafe and violate the safety constraint. For the remaining states, the optimization is solved via binary search over z. The resulting state-feedback policy for agent i, $\pi^i_\theta : \mathcal{X} \times [t,T) \to \mathcal{U}$, is given by

$$\pi^i_\theta(t,x) = \arg\min_{\mathbf{u}_d} \langle \nabla \hat{V}_\theta(t,\hat{o}^*), \ \hat{f}_d(\hat{o}^*, \mathbf{u}_d) \rangle,$$

where $\hat{o}^* = [o, z^*]^T$ is the augmented state corresponding to the optimal z^*.

Since interaction structures in multi-agent navigation evolve dynamically, the neighbour set $\mathcal{N}_i$ is updated online, motivating a receding-horizon execution of the

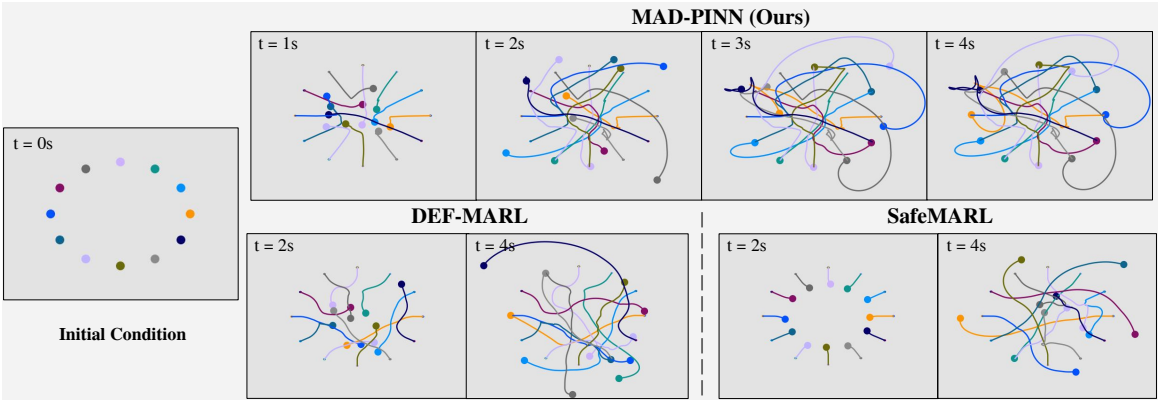


Figure 9.3: Snapshots of multi-agent navigation trajectories at different times using MAD-PINN and baselines. Agents are represented as circles with radius R, indicating the minimum safe distance they must maintain from each other. Smaller dots mark their respective goals. MAD-PINN trajectories show that agents proactively **maintain long-horizon safety** by adjusting their paths to avoid close encounters, rather than enforcing safety reactively (SafeMARL), which could lead to suboptimal behaviours.

policy. This ensures that policy continuously adapts by incorporating updated interaction information and anticipating future conflicts. In addition, frequent re-planning further enhances robustness to model mismatch, sensor noise, and external disturbances, enabling reliable decentralized navigation. Finally, this framework naturally extends to long-horizon tasks by repeatedly solving the SC-OCP over shorter horizons, allowing the proposed framework to co-optimize safety and performance in a practical and computationally efficient manner for real-world autonomous systems.

9.5 Experiments

	3 Agents		8 Agents		12 Agents		16 Agents	
Method	Safety	Safe Sc.	Safety	Safe Sc.	Safety	Safe Sc.	Safety	Safe Sc.
Ours	100% ± 0.0%	100% ± 0.0%	99.5% ± 0.4%	98% ± 1.2%	99.3% ± 0.6%	96% ± 1.8%	98.3% ± 0.8%	86% ± 2.7%
DEF-MARL (3)	94% ± 2.8%	91% ± 3.4%	89.8% ± 2.5%	63% ± 4.2%	71% ± 3.8%	31% ± 4.8%	44.3% ± 3.4%	7% ± 1.5%
DEF-MARL (8)	100% ± 0.0%	100% ± 0.0%	95.4% ± 2.1%	82% ± 3.2%	75.2% ± 3.5%	37% ± 4.5%	50.9% ± 2.9%	13% ± 2.1%
SafeMARL	100% ± 0.0%	100% ± 0.0%	99.8% ± 0.2%	99% ± 0.8%	99% ± 0.5%	96% ± 1.7%	97.5% ± 1.0%	90% ± 2.2%
MPPI	98% ± 1.1%	97% ± 1.4%	89.5% ± 2.8%	72% ± 3.5%	72.9% ± 3.7%	35% ± 4.9%	48.8% ± 3.6%	9% ± 2.4%

Table 9.1: Average safety rates and safe-scenario percentages across varying numbers of agents. Our method maintains consistently high values, indicating collision-free execution of every agent for nearly all initial configurations and demonstrating **effective safety-performance co-optimization at the agent and system level**.

The goal of our experiments is to assess the effectiveness of the proposed framework in (i) co-optimizing safety and performance in multi-agent systems, (ii) scaling to larger environments with higher numbers of agents, and (iii) validating the proposed safety-aware neighbour selection strategy.

9.5.1 Baselines

To thoroughly assess our method, we benchmark it against baselines that represent the spectrum of safety integration techniques: 1) *Constrained Policy Synthesis:* **DEF-MARL** [201], which uses multi-agent RL (MARL) to solve the discrete-time epigraph reformulation of the SC-OCP to synthesize safe and optimal policies; 2) *Safety Filtering:* **SafeMARL** [32], which uses a control barrier value function (CBVF)-based safety filter to provide safety for a nominal MARL policy; and 3) *Soft-Constrained Optimization:* **MPPI** [185], a sampling-based MPC method to solve (9.4) that penalizes constraint violations in its cost function using a Lagrangian approach.

9.5.2 Evaluation Metrics

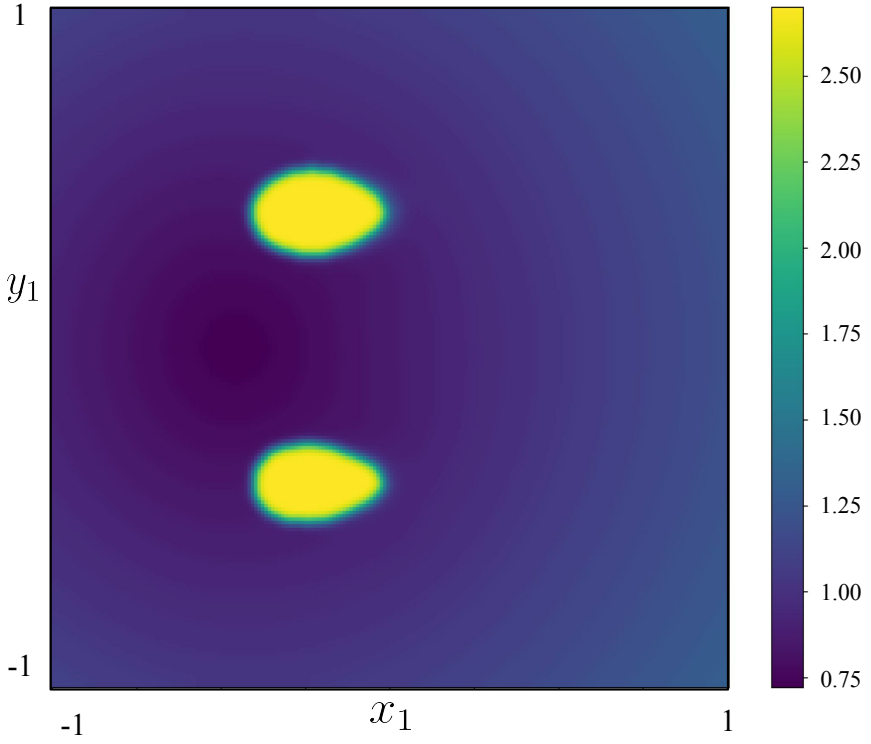


Figure 9.4: **Heatmap of the learned value function with respect to the ego agent's position coordinates.** The other two agents are at $[-0.3, 0.4]$ and $[-0.3, -0.4]$, both moving with velocity $[-1, 0]$, while the ego agent moves with velocity $[1, 0]$ toward its goal at $[-0.5, 0]$.

To evaluate the trade-off between performance and safety, we use the following metrics: **(1) Cumulative Cost:** The total cost $\int_0^T l(x(s))ds + \varphi(x(T))$ accumulated along safe trajectories. **(2) Safety Rate:** The fraction of agents that remain collision-free for the entire horizon, quantifying per-agent safety. **(3) Safe Scenarios:** The fraction of scenarios in which all agents remain collision-free for the entire horizon, reflecting collective safety guarantees.

Number of Agents	Cumulative Cost	Safety Rate	Safe Scenarios
64	1.09	100%	100%
128	1.68	98.75%	95%
256	2.73	96.25%	85%

Table 9.2: Effect of increasing agent count and environment size. Our method **co-optimizes safety and performance even in large multi-agent environments**, despite training on smaller environments with fewer agents.

9.5.3 Experimental Setup

We study a multi-agent **Drone Navigation** problem where each agent follows double-integrator dynamics with state $\mathbf{s}_i = [x, y, v_x, v_y]^\top$, constrained to $(x, y) \in [-1, 1]^2$ and $(v_x, v_y) \in [-4, 4]^2$. The control input is $\mathbf{u}_i = [a_x, a_y]^\top \in [-4, 4]^2$, representing acceleration. Each agent is assigned a parameterized goal (x_g, y_g), and its dynamics are given by $\dot{x} = v_x$, $\dot{y} = v_y$, $\dot{v}_x = a_x$, $\dot{v}_y = a_y$. Agents must reach their goals while maintaining a safety distance $r = 0.1$. The running cost for training the decentralized value function is:

$$l_d = \sum_{j=1}^{|\mathcal{N}_i|} \|(x^j(t), y^j(t))^\top - (x_g^j(t), y_g^j(t))^\top\|, \tag{9.13}$$

where $\mathcal{N}_i$ denotes the neighbor set. We train the auxiliary value function with 3 agents (so $|\mathcal{N}_i| = 2$), a time horizon of 0.2s, and an observation radius $r_{obs} = 0.5$, and deploy the same value function for all agents across all environments. The residual component of the auxiliary value function R_θ is approximated as a multi-layer perceptron (MLP) with three hidden layers of 256 neurons each, using sine activation functions. The network is trained with the Adam optimizer at a learning rate of 2×10^{-5}. We evaluate all algorithms over 100 distinct initial conditions across 5 seeds.

Figure 9.4 illustrates the heatmap of the learned value function as a function of the ego agent's position coordinates. Regions of high value (yellow) correspond to unsafe states overlapping with the positions of other agents, thus encoding safety constraints. Conversely, regions of low value (dark purple) are concentrated around the goal, reflecting the task objective. This demonstrates that the value function integrates both safety and performance considerations, enabling the derived policy to effectively co-optimize these objectives.

9.5.4 Results

1) Effectiveness in Co-optimizing Safety and Performance: To gauge the ability of our method in co-optimizing safety and performance, we compare it with the baselines on 3, 8, 12, and 16 agents, with environment size fixed so that density increases as the number of agents grows. As shown in Figure 9.2, our method consistently achieves the best trade-off, attaining the lowest cumulative cost while maintaining near-perfect safety rates, thereby validating its ability to co-optimize both objectives. In contrast, DEF-MARL, trained with 3 and 8 agents, performs competitively when evaluated on the same agent counts but suffers pronounced degradation in both safety and performance as agent density increases, highlighting the limited generalization of RL-based co-optimization approaches. Safety-filtering baselines such as SafeMARL effectively prevent collisions, but their overly conservative behavior results in substantially higher costs. Penalty-based methods such as MPPI fare even worse, exhibiting both high costs and poor safety rates due to the soft enforcement of safety constraints. Table 9.1 further illustrates these differences through the stringent *safe scenarios* metric, which requires collision-free execution across all agents for the entire horizon. Our method consistently achieves a very high percentage of safe scenarios, indicating not only strong safety rates but also collision-free execution in nearly all cases. This highlights the effectiveness of decentralization, where each agent can co-optimize safety and performance individually. By contrast, non-safety filtering baselines such as DEF-MARL and MPPI, while achieving moderate safety rates, perform poorly on the safe scenarios metric as the number of agents grows, suggesting their inability to co-optimize safety and performance at the agent level. SafeMARL, while maintaining high safe-scenario rates, does so at the cost of severely degraded performance. Taken together, these results show that our approach inherits the strengths of safety-filtering methods (high safety) and performance-driven methods (low cost), while avoiding their drawbacks. The performance degradation of our approach remains minimal when scaling from 3 to 16 agents despite the increase in density, demonstrating that it can effectively co-optimize safety and performance using only local observations; a property that makes it well-suited for real-world autonomous systems, where agents typically have access only to local state information. Figure 9.3 further substantiates our claims by demonstrating that the proposed method guarantees long-horizon safety while allowing all agents to successfully reach their respective goals without collisions. In contrast, baseline approaches either adopt reactive safety strategies (leading to over-conservatism) or fail to preserve safety, highlighting their limitations in co-optimizing safety and performance.

2) Scalability with Agent Count and Environment Size: To evaluate the scalability of our approach, we conduct experiments with 20 distinct initial conditions and a substantially larger number of agents, namely 64, 128, and 256. In addition,

Method	Cumulative Cost	Safety Rate	Safe Scenarios
Value-based (Ours)	0.51	99.33%	96%
Nearest	0.82	83%	45%
Random	1.55	33%	4%

Table 9.3: Ablation study on the impact of the proposed neighbor selection strategy. By effectively **capturing safety-critical inter-agent interactions among neighbors**, it enables improved safety–performance co-optimization.

the environment size is increased from $[-1, 1]^2$ to $[-4, 4]^2$ to test the method in a more extensive and populated setting compared to the training environment. From Table 9.2, it can be observed that our approach consistently achieves very high safety rates even in large environments with up to 256 agents, while maintaining a high safe-scenario percentage. This indicates that each agent is able to co-optimize safety and performance effectively, even as the number of agents increases. The safety rates and safe-scenario percentages remain comparable across Tables 9.1 and 9.2 because the increased environment size in this study results in a similar agent density in both setups. The cumulative costs are higher in this setting, as agents must travel longer distances to reach their respective goals due to the environment being scaled by a factor of 4 in each dimension. However, the increase in cost is approximately proportional to the increase in environment dimensions, suggesting that our method preserves the same level of performance as in Table 9.1. These findings further confirm that, within a decentralized setting, our method can effectively address extremely large-scale multi-agent problems by training policies using only local information in a smaller environment and subsequently applying the same policy independently to each agent. This highlights the practicality of our framework for deployment in large-scale multi-agent systems.

3) Effectiveness of Neighbour Selection: Finally, we conduct an ablation study to evaluate the effect of the proposed HJ Reachability-based neighbour selection strategy. We compare it against two alternative strategies: (1) a distance-based strategy, where each agent selects its two nearest neighbours, and (2) a random strategy, where each agent selects two neighbours uniformly at random within its observation radius. All strategies are evaluated on the same 12-agent environment used in Table 9.1. As shown in Table 9.3, our method consistently achieves higher safety rates and lower costs, indicating that prioritizing neighbours based on the pairwise safety value functions V_s effectively captures the most critical interactions. In contrast, the distance-based strategy leads to a myopic neighbour selection, as it ignores safety-critical information (e.g., agents' relative velocities), thereby reducing safety rates and drastically lowering the safe scenario fraction. The random selection strategy performs the worst, as it fails to incorporate any information about the surrounding agents. These results

highlight the necessity of a principled neighbour selection framework that accounts for critical inter-agent interaction information to ensure reliable decentralized multi-agent navigation.

9.6 Chapter Summary

We presented a physics-informed machine learning framework for scalable multi-agent safe and optimal control. The key contribution lies in the reformulation of the large-scale SC-OCP into a decentralized formulation with fixed observation size, enabling tractability while preserving global consistency. To address the curse of dimensionality in solving the associated HJB-PDE, we introduced a principled neighbour selection strategy based on reachability analysis, ensuring that agents account for the most safety-critical interactions. Extensive experiments demonstrated that our approach effectively co-optimizes safety and performance, scales favourably with the number of agents, and significantly outperforms existing performance-driven and safety-driven baselines.

Future work will focus on extending the framework to heterogeneous agent systems, incorporating dynamics and model uncertainty, and evaluating performance on real-world robotic platforms. We believe this work represents a step toward practical, safe, and scalable multi-agent autonomy. In addition, we plan to quantify the approximation error of the auxiliary value function [95, 158, 160] and investigate its impact on both safety and performance.

Bridge to the Next Chapter. Chapters 6–9 have developed safety and performance frameworks that operate on *low-dimensional state representations*—positions, velocities, and orientations assumed to be directly observable. In practice, however, robots perceive their environments through *high-dimensional sensory observations* such as camera images. The final technical chapter of this dissertation addresses this perception gap by extending barrier-certificate-based safety to visuomotor control, where a safe policy must be synthesized directly from raw image inputs.

Part IV

Safety in Latent Spaces

Bringing safety-critical control into the real world, this part lifts barrier certificate synthesis from full-state feedback to raw visual observations, learning safe visuomotor policies in latent representation spaces.

Chapter 10

Safe Visuomotor Control with Barrier Certificates

10.1 Related Works

The preceding chapters developed increasingly powerful safety frameworks, from classical CBFs (Chapter 6) through neural barriers with formal verification (Chapters 3–4), to co-optimization of safety and performance (Chapters 7–9). However, all these methods operate on *low-dimensional state representations*—positions, velocities, and orientations that are assumed to be directly observable. In many practical scenarios, the robot perceives its environment through *high-dimensional observations* such as camera images, and the relevant state must be inferred from these observations. This chapter addresses the question: *can we maintain formal safety guarantees when the controller operates directly from visual inputs?*

Visuomotor Policy Learning. End-to-end visuomotor learning maps raw sensory inputs directly to control actions using deep neural networks [90, 80]. While expressive and flexible, these policies provide no formal safety guarantees. Extending the vision-based safety direction developed in this chapter, subsequent work by the thesis author studies feasibility-aware safety alignment for vision–language–action policies [157]. Sim-to-real approaches [172] and self-supervised learning [119] improve robustness but do not address the fundamental issue of constraint satisfaction.

Safe Control from Perception. Recent works have begun to integrate safety constraints with perception-based control. CBF-based approaches that use learned perception modules [47] maintain safety in the physical state space but require access

to a state estimator with bounded error. Latent-space safety methods [173] learn barrier functions in the representation space of a variational autoencoder but lack formal completeness guarantees; the latent barrier may not correspond to actual safety in the physical world.

Control Barrier Certificates. For discrete-time systems, Control Barrier Certificates (CBCs) [73] provide an analogous safety framework to CBFs. Neural CBCs [1] learn these certificates from data, and completeness results establish conditions under which the learned certificate provably exists and correctly separates safe from unsafe states. However, existing neural CBC methods operate on state-based inputs and have not been extended to high-dimensional observations.

Gap Addressed. This chapter bridges the gap between visuomotor learning and safety certification by constructing CBCs in a learned latent space. The key innovation is the Safety-based Latent Dynamics (SaLaD) model, which learns a latent representation aligned with the safety labels of the physical system. Combined with a semi-supervised policy synthesis procedure, this yields coverage-relative latent-space certificates under the stated cover, Lipschitz, and latent-dynamics fidelity assumptions.

10.2 Contributions of this Chapter

This chapter makes the following contributions:

1. We introduce the *Safety-based Latent Dynamics (SaLaD)* model, which learns a compact latent-space representation from high-dimensional observations while preserving the safety-relevant structure needed for barrier certificate construction.

2. We synthesize a *Neural Control Barrier Certificate* in the latent space alongside a policy via semi-supervised learning. Under the stated cover, Lipschitz, and latent-dynamics fidelity assumptions, the learned CBC satisfies a completeness condition without post-hoc pointwise verification.

3. We derive a *consistency condition* for non-zero dynamics that bounds violation of the CBC decrease condition on the training samples. Extending this statement throughout the covered latent region additionally requires a bound on latent-dynamics error over that region.

4. We validate the framework on an inverted pendulum stabilization task from pixel observations and a quadruped robot navigation task, demonstrating safe visuomotor control from raw perception that outperforms Imitation Learning and PPO-Lagrangian baselines.

The results presented in this chapter are based on [159]. Additional material is available on the project webpage[1], and the accompanying code is available online[2].

10.3 Preliminaries and Problem Formulation

In this section, we formally introduce Control Barrier Certificates (CBCs), existing work on providing completeness guarantees for Neural CBCs, and finally, the problem formulation for safe visuomotor control.

10.3.1 Notations

For a set A, we define the indicator function of A, denoted by $\mathbb{1}_A(x)$, where, $(\mathbb{1}_A(x)) = 1$ when $x \in A$ and 0 otherwise. The complement of a set A within a set B is denoted by $B \backslash A$.

10.3.2 System Description

A discrete-time control system is a tuple $\mathbb{S} = (X, U, f)$, where $X \subseteq \mathbb{R}^n$ is the state set of the system, $U \subseteq \mathbb{R}^{n_u}$ is the input set of the system, and $f : X \times U \to X$ describes the state evolution of the system via the following difference equation:

$$x(t+1) = f(x(t), u(t)), \forall t \in \mathbb{N}, \tag{10.1}$$

where $x(t) \in X$ and $u(t) \in U$, $\forall t \in \mathbb{N}$, denote the state and input of the system, respectively.

Consider a set $\mathcal{C}$ defined as the *sub-zero level set* of a continuous function $B : X \subseteq \mathbb{R}^n \to \mathbb{R}$ yielding,

$$\mathcal{C} = \{x \in X \subset \mathbb{R}^n : B(x) \leq 0\} \tag{10.2}$$

$$X - \mathcal{C} = \{x \in X \subset \mathbb{R}^n : B(x) > 0\}. \tag{10.3}$$

[1] https://tayalmanan28.github.io/vision_cbf/
[2] https://github.com/tayalmanan28/vision_cbf

We further restrict the class of $\mathcal{C}$ where its interior and boundary are precisely the sets given by $\text{Int}(\mathcal{C}) = \{x \in X \subset \mathbb{R}^n : B(x) < 0\}$ and $\partial\mathcal{C} = \{x \in X \subset \mathbb{R}^n : B(x) = 0\}$, respectively.

10.3.3 Control Barrier Certificates (CBCs)

In this section, we introduce the notion of a control barrier certificate, which provides sufficient conditions together with controllers for the satisfaction of safety constraints.

Definition 10.3.1. A function $B : X \to \mathbb{R}$ is a control barrier certificate for a discrete-time control system $\mathbb{S} = (X, U, f)$ if for any state $x \in X$ there exists an input $u \in U$, such that

$$B(f(x,u)) \leq B(x), \tag{10.4}$$

The following lemma allows us to synthesize controllers for discrete-time control system $\mathbb{S}$, ensuring the satisfaction of safety properties.

Lemma 10.1 ([74]). *For a discrete-time control system $\mathbb{S} = (X, U, f)$, safe set $X_s \subseteq \mathcal{C}$, and unsafe set $X_u \subseteq X - \mathcal{C}$, the existence of a control barrier certificate, B, as defined in Definition 10.3.1, under a control policy $\pi : X \to U$ implies that the sequence state in $\mathbb{S}$ starting from $x_s \in X_s$ under the policy π do not reach any unsafe states in X_u.*

The zero-level set of the CBC $B(x) = 0$ separates the unsafe regions from the safe ones. For an initial state x_0 within the safe region ($x_0 \in X_s$), $B(x_0) \leq 0$ holds by construction of the safe region. According to equation (10.4), which ensures $B(x)$ remains non-increasing, the level set is not crossed, preventing access to unsafe regions. Therefore, ensuring system safety requires computing appropriate control barrier certificates and corresponding control policies.

10.3.4 Neural CBCs with Completeness Guarantees

To extend sampled CBC inequalities beyond the finite training set, we consider a specified latent region $X_{\text{cov}} \subseteq X$ covered by the samples. A validity condition adapted from [165, 9] propagates the sampled inequalities over this covered region under a Lipschitz assumption. Specifically, we introduce a positive scalar margin, ψ, to the inequalities (2.8) and (10.4), as follows:

$$q_i(x) + \psi \leq 0, \forall i \in 1, 2, 3, \tag{10.5}$$

where $q_1(x) = (B(x))\mathbb{1}_{X_s}$, $q_2(x) = (-B(x))\mathbb{1}_{X_u}$ and $q_3(x) = B(f(x,u)) - B(x)$. The following theorem provides a theoretical lower bound of ψ.

Theorem 10.3.2. *Consider a discrete-time robotic system and initial safe and unsafe sets $X_s, X_u \subseteq X$, respectively. Let $X_{\text{cov}} \subseteq X$ be a specified latent region and suppose that N samples x_i form an $\bar{\epsilon}$-cover of it: for every $x \in X_{\text{cov}}$, there exists an x_i such that $\|x - x_i\| \leq \bar{\epsilon}$. Let B_θ be the neural network-based CBC, and assume the corresponding sampled CBC residuals are Lipschitz on X_{cov} with constant L_B. Then the sampled CBC inequalities extend over X_{cov} if the following condition holds:*

$$\psi^* \geq L_B \bar{\epsilon}, \tag{10.6}$$

where L_B is the Lipschitz constant of B_θ.

Remark 10.3.3 (Scope of the Guarantee and Latent-to-Physical Safety). Two clarifications regarding Theorem 10.3.2 are in order. *(i) Scope.* The validity condition $\psi^* \geq L_B\bar{\epsilon}$ certifies the CBC condition over the entire state space *covered by the $\bar{\epsilon}$-cover*—not merely at the training samples—provided the Lipschitz bound L_B holds on that covered region. What remains sample-dependent is the fidelity with which the learned latent dynamics reproduce the true encoded dynamics outside the support of the data; the certificate inherits exactly this fidelity. *(ii) From latent to physical safety.* In this chapter, X is the learned latent space. The SaLaD objective in Section 10.4.1 combines safe-label and unsafe-label barrier losses with a latent-state consistency loss. Safe and unsafe labels are generated from task-specific safety checks on simulator states—such as angle limits for the pendulum and collision checks for navigation—and mapped into latent space through the encoder. At deployment, incoming images pass through the same encoder, so correspondence between latent and physical safety remains conditional on the encoder retaining the safety-relevant information and on the scene lying within the represented configuration distribution. Perception aliasing, lighting shifts, or out-of-distribution scenes are outside the certificate. Regarding information content, compressing high-dimensional observations into a compact latent code can discard safety-relevant detail and thereby introduce conservatism or blind spots. The safe/unsafe barrier losses encourage alignment with the labeled partition, and the experiments compare the resulting policy with GCIL, PPO-Lagrangian, and an unverified SaLaD ablation. They do not constitute a formal proof that all safety-relevant physical information is preserved; characterizing the information loss and induced conservatism remains an open problem.

10.3.5 Problem Formulation

extitGiven a discrete-time robotic system $\mathbb{S}$ as defined in (10.1). Let $\mathcal{S}$ represent samples of visuomotor observations from the safe region, $\mathcal{U}$ represent samples from the unsafe region, and $\mathcal{D}$ denote the complete set of samples (both labeled and unlabeled). The objective is to devise an algorithm to jointly synthesize a parameterised barrier certificate B_θ and policy π_θ satisfying (10.4) on the covered latent region, subject to the stated cover, Lipschitz, and learned-dynamics assumptions.

In the following section, we propose an algorithmic approach to solve the above problem.

10.4 Semi-Supervised Policy Learning Framework

We introduce a semi-supervised policy learning framework that employs control barrier certificates (CBCs) to jointly learn both the barrier certificate and a provably safe policy. Specifically, this framework uses the forward invariance properties of CBCs to ensure that the learned policy remains within the safe set, provided the initial conditions are within this safe set. However, due to the absence of a barrier certificate, we lack access to the safe set $\mathcal{C}$. Hence, we start with initial safe and unsafe sets $X_s \subseteq \mathcal{C}$ and $X_u \subseteq X - \mathcal{C}$, respectively, such that any trajectory starting in X_s never enters X_u, which makes this framework semi-supervised. We collect the data sets $\mathcal{S}$,$\mathcal{U}$ and $\mathcal{D}$ corresponding to N visuomotor data points sampled from the initial safe set $X_s \subseteq \mathcal{C}$, initial unsafe set $X_u \subseteq X - \mathcal{C}$, and state set X, respectively.

We represent the barrier functions and the policy as neural networks, denoted as B_θ and π_θ respectively. These functions are learned in a latent space (z), encoded by representation network E_θ and use the learned (latent) dynamics d_θ, given by the SaLaD model (Subsection (10.4.1)). For notational simplicity, we denote parameters of all the above mentioned neural networks as θ (online) and θ^- (target; slow-moving average of θ) as combined feature vectors.

10.4.1 Safety based Latent Dynamics (SaLaD)

Separability of safe and unsafe regions in the feature space is essential for learning an accurate barrier certificate. However, in real space, the boundary separating safe and unsafe regions may be complex, which can pose challenges in accurately learning the barrier function. To mitigate this, we propose **SaLaD**, a **Sa**fety-driven **La**tent

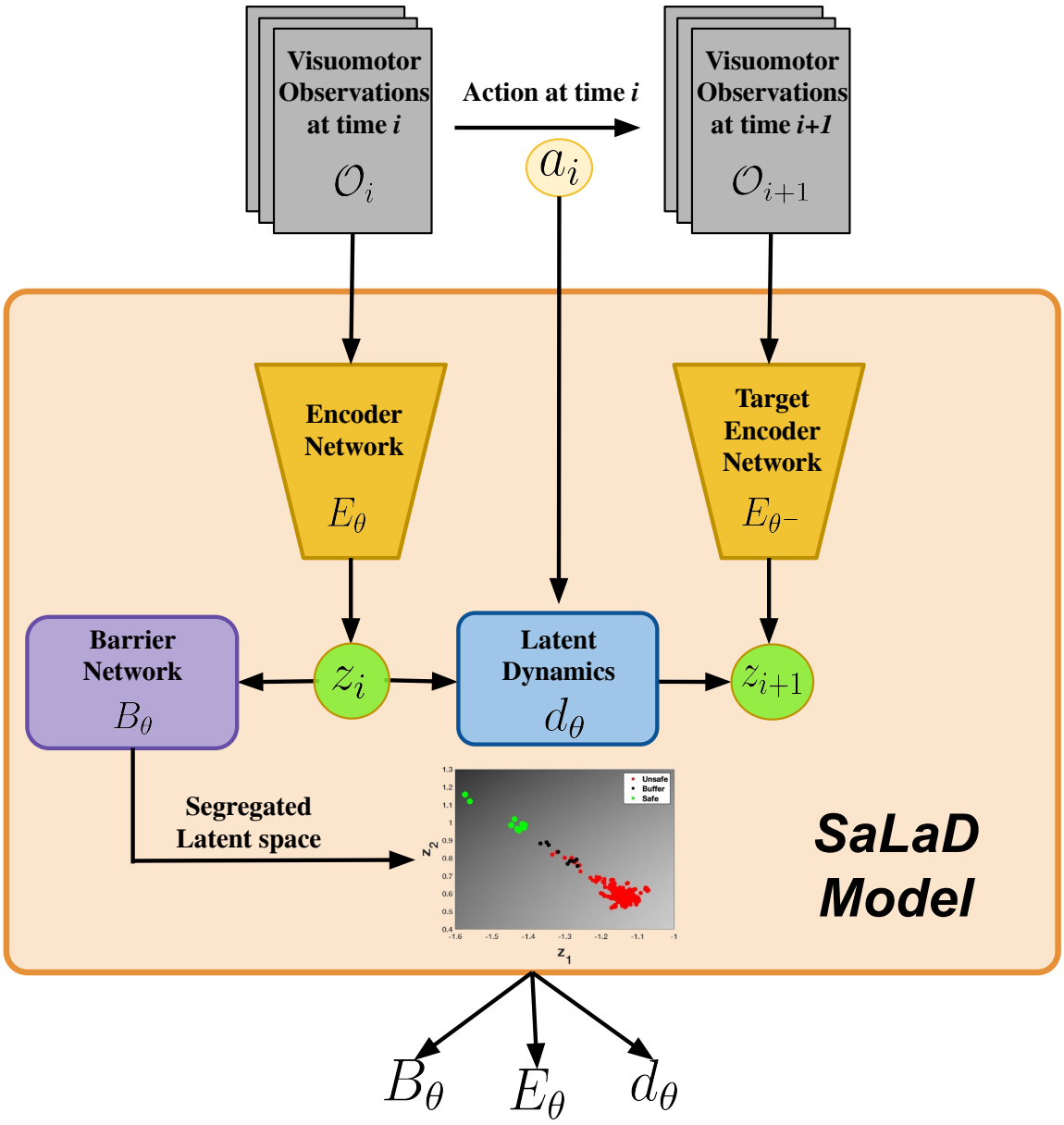


Figure 10.1: Presents the architecture of the Safety-driven Latent Dynamics (SaLaD) model. **Training the SaLaD model** the visuomotor observation $\mathcal{O}_i$ is encoded by E_θ into a latent representation z_i. Then, SaLaD predicts the next latent states z_{i+1} as well as the Barrier Certificate over the safe and unsafe datasets for each latent state, and we optimize SaLaD using Equation (10.7). Next, visuomotor observation $\mathcal{O}_{i+1}$ is encoded using target net E_{θ^-} and used as latent targets only during training.

Dynamics model that learns a latent representation of the environment where the safe and unsafe regions are sufficiently separated while maintaining the state consistency in the latent space. We achieve this by only modelling the elements of the environment that help in segregating the latent space into safe and unsafe regions rather than attempting to model the environment itself.

Given an observation $\mathcal{O}_i$ observed at time-step i, the network E_θ encodes $\mathcal{O}_i$ into a latent representation z_i: $z_i = E_\theta$. From z_i and a random action a_i taken at time-step i, SaLaD then predicts the latent dynamics d_θ (latent representation z_{i+1} of the following time-step). Given the safe and unsafe datasets $\mathcal{S}$,$\mathcal{U}$ respectively, we learn the barrier certificate B_θ, that helps in segregating safe and unsafe regions. Now, let us consider the following loss functions satisfying the conditions over the training data sets $\mathcal{S}$,$\mathcal{U}$,$\mathcal{D}$ as follows:

$$\begin{aligned}\mathcal{L}_{SaLaD}(\theta) =& \xi_1 \sum_{z_i \in \mathcal{S}} \max\left(0, B_\theta(z_i) + \psi\right) + \\ & \xi_2 \sum_{z_i \in \mathcal{U}} \max\left(0, -B_\theta(z_i) + \psi\right) + \\ & \xi_3 \sum_{z_i \in \mathcal{D}} \left(E_{\theta^-}(\mathcal{O}_{i+1}) - d_\theta(z_i, a_i)\right)^2,\end{aligned} \tag{10.7}$$

where d_θ represents the latent space dynamics network parameterized by θ. The first term in loss function $\mathcal{L}_{SaLaD}$ represents the loss for safe states, second term represents the loss for unsafe states, respectively. The third term in $\mathcal{L}_{SaLaD}$ represents the latent state consistency loss. ξ_1, ξ_2 and ξ_3 represents the positive relative weights of each of the losses. $\psi > \psi^*$ is a scalar margin introduced to satisfy the stated covered-region completeness condition, where ψ^* is given by Eq. (10.6).

10.4.2 Formal Verification Integration

In accordance with Theorem (10.3.2), it is necessary to establish a bound on the neural network's Lipschitz constant to ensure completeness guarantees. To train neural networks with Lipschitz bounds, we have the following lemma:

The log-determinant barrier is evaluated only on the positive-definite domain of M; maintaining a finite barrier value therefore preserves this domain during optimization. The numerical value of the log-determinant loss alone is not a separate certificate of positive semidefiniteness. The loss functions characterizing the satisfaction of Lipschitz bound is given as:

$$\mathcal{L}_M(\theta, \Lambda) = -\log\det(M(\theta, \Lambda)), \tag{10.8}$$

Algorithm 13 Semi-Supervised Safe Policy Synthesis

Require: Inputs: Observation data sets $\mathcal{S},\mathcal{U},\mathcal{D}$, Lipschitz bound L_B, warm-start iterations $iter_{warmstart}$, loss weights $\lambda_1, \lambda_2, \lambda_3$

Ensure: Outputs: Trained encoder E_θ, latent dynamics d_θ, barrier certificate B_θ, safe visuomotor policy π_θ

Initialise(θ, ψ, Λ)
$\mathcal{O}_i \leftarrow sample(\mathcal{S},\mathcal{U},\mathcal{D})$
for $i = 1$ to $iter_{warmstart}$ **do**
  $\psi \leftarrow L_B\bar{\epsilon}$ ▷ From eq. (10.6)
  $\mathcal{L}_{SaLaD} \leftarrow (B_\theta, E_\theta, d_\theta, \mathcal{O}_i, a_i, \psi)$ ▷ From eq. (10.7)
  $\theta \leftarrow Learn(\mathcal{L}_{SaLaD}(\theta), \theta)$
  $B_\theta, E_\theta, d_\theta \leftarrow \theta$
  $\bar{\epsilon} \leftarrow (E_\theta, \mathcal{D})$

$\mathcal{L}_{total}(\theta) = \lambda_1\mathcal{L}_{syn}(\theta) + \lambda_2\mathcal{L}_{SaLaD}(\theta) + \lambda_3\mathcal{L}_{\pi}(\theta)$
▷ From eq. (10.7),(10.9),(10.12)

while $\mathcal{L}_{total} > 0$ or $\mathcal{L}_M \not\leq 0$ **do**
  $\psi \leftarrow L_B\bar{\epsilon}$ ▷ From eq. (10.6)
  $\eta \leftarrow L_B\delta$ ▷ From eq. (10.10)
  $\mathcal{L}_{total} \leftarrow (B_\theta, E_\theta, d_\theta, \pi_\theta, \mathcal{O}_i, \psi, \eta)$
  $\theta \leftarrow Learn(\mathcal{L}_{total}(\theta), \theta)$
  $\mathcal{L}_M \leftarrow (\theta, \Lambda)$ ▷ From eq. (3.9)
  $\theta, \Lambda \leftarrow Learn(\mathcal{L}_M)$
  $B_\theta, \pi_\theta, E_\theta, d_\theta \leftarrow \theta$
  $\mathcal{D} \leftarrow \pi_\theta$ (Rollouts)
  $\bar{\epsilon} \leftarrow (E_\theta, \mathcal{D})$
  $\delta \leftarrow (E_\theta, d_\theta, \mathcal{D})$
end while

where M is the semi-definite matrix corresponding to the Lipschitz bounds L_B and Λ is a trainable parameter.

10.4.3 Controller Synthesis

In this subsection, we will discuss the synthesis of a safe policy. Based on Lemma 1, satisfying condition (10.4) ensures that the synthesized policy prevents trajectories starting in the safe set from entering unsafe regions while also ensuring that trajectories originating in unsafe regions eventually reach the safe set. The loss function corresponding to the satisfaction of condition (10.4) is formulated as follows:

$$\mathcal{L}_{syn}(\theta) = \sum_{x_i \in \mathcal{D}} \max(0, B_{\theta^-}(d_\theta(E_{\theta^-}(\mathcal{O}_i), \pi_\theta(z_i))) + \eta - B_\theta(z_i) + \psi), \tag{10.9}$$

where $\eta > \eta^*$ is a scalar value introduced to compensate for the error in learned (latent) dynamics. The following theorem provides a theoretical lower bound on the

value of η for the training samples:

Theorem 10.4.1. *Consider a discrete-time robotic system and the (latent) dynamics learned using the SaLaD model. Let δ represent the maximum latent state consistency error for the training samples. Then the condition (10.4) is valid on the training samples, if the following condition holds:*

$$\eta^* \geq L_B \delta, \tag{10.10}$$

where L_B is the Lipschitz constant of B_θ and

$$\delta = \max_{i \in \{1,\ldots,N\}} \|d_\theta(z_i, u) - E_\theta(\mathcal{O}_{i+1})\|_2. \tag{10.11}$$

Improving the performance of the synthesized controller: Since CBCs only guarantee safety and not performance, we introduce the following loss term to ensure that the synthesized policy, π_θ, enhances the safety of the user-defined policy, π_{user} (can be an LQR or PID controller), while remaining close to baseline performance.

$$\mathcal{L}_\pi(\theta) = \|\pi_\theta - \pi_{user}\|_2. \tag{10.12}$$

10.4.4 Training Scheme

The training procedure is divided into the following stages:

Warm Starting the SaLaD Model

To initialize our training process, we use a warm start approach by optimizing the SaLaD model with the loss function $\mathcal{L}_{SaLaD}$ on samples generated from uniform random actions over a set number of epochs. This method reduces initial model biases, leading to more stable and efficient learning. It also accelerates convergence by offering a well-informed starting point, minimizing the risk of poor local minima, and enhancing overall model performance through more balanced action space exploration.

Iterative Training with Controller Synthesis

During the training process, we follow an iterative approach consisting of four key steps:

1. Collecting data through roll-outs from the policy π_θ.

2. Improving the policy by optimizing the loss functions $\mathcal{L}_{syn}$ and $\mathcal{L}_{\pi}$.

3. Fine-tuning the SaLaD model over newly collected data.

4. Optimizing $\mathcal{L}_M$ to ensure that the learned barrier function maintains Lipschitz continuity.

This iterative procedure helps in progressively refining the policy and model, thereby improving overall performance and stability. The overall algorithm is summarised in Algorithm (13).

10.5 Case Studies

In this section, we assess the efficacy of our proposed framework through two distinct case studies: the inverted pendulum system and the obstacle avoidance of an autonomous mobile robot. Both case studies are conducted on a computing platform equipped with an Intel i9-11900K CPU, 32GB RAM, and NVIDIA GeForce RTX 4090 GPU.

Baselines: We evaluate our method against two baselines: Goal Conditioned Imitation Learning (GCIL) [48], which learns policies through behavior cloning under randomized safety conditions without explicit conditioning on safety parameters, relying on domain randomization for generalization; and PPO-Lagrangian, a constrained reinforcement learning (CRL) approach based on Proximal Policy Optimization (PPO) [137, 130], where safety constraints are incorporated as soft penalties within a Lagrangian framework. Additionally, we conduct an ablation study by comparing against a variant of SaLaD that omits formal verification during training (as detailed in Sec. 10.4.2), allowing us to assess the impact of integrating formal verification on policy synthesis.

10.5.1 Inverted Pendulum

In our first case study, we analyze an inverted pendulum system characterized by the state vector $x = [\Theta, \dot{\Theta}] \in [-\pi, \pi] \times [-3.5, 3.5]$, where Θ represents the angular position, and $\dot{\Theta}$ denotes the angular velocity. The visuomotor observation $\mathcal{O}_t$ consists of a stack of two RGB frames, each with dimensions of 64×64 pixels. The control input to the system is the applied torque $\tau \in [-10, 10]$ Nm, as illustrated in Fig.

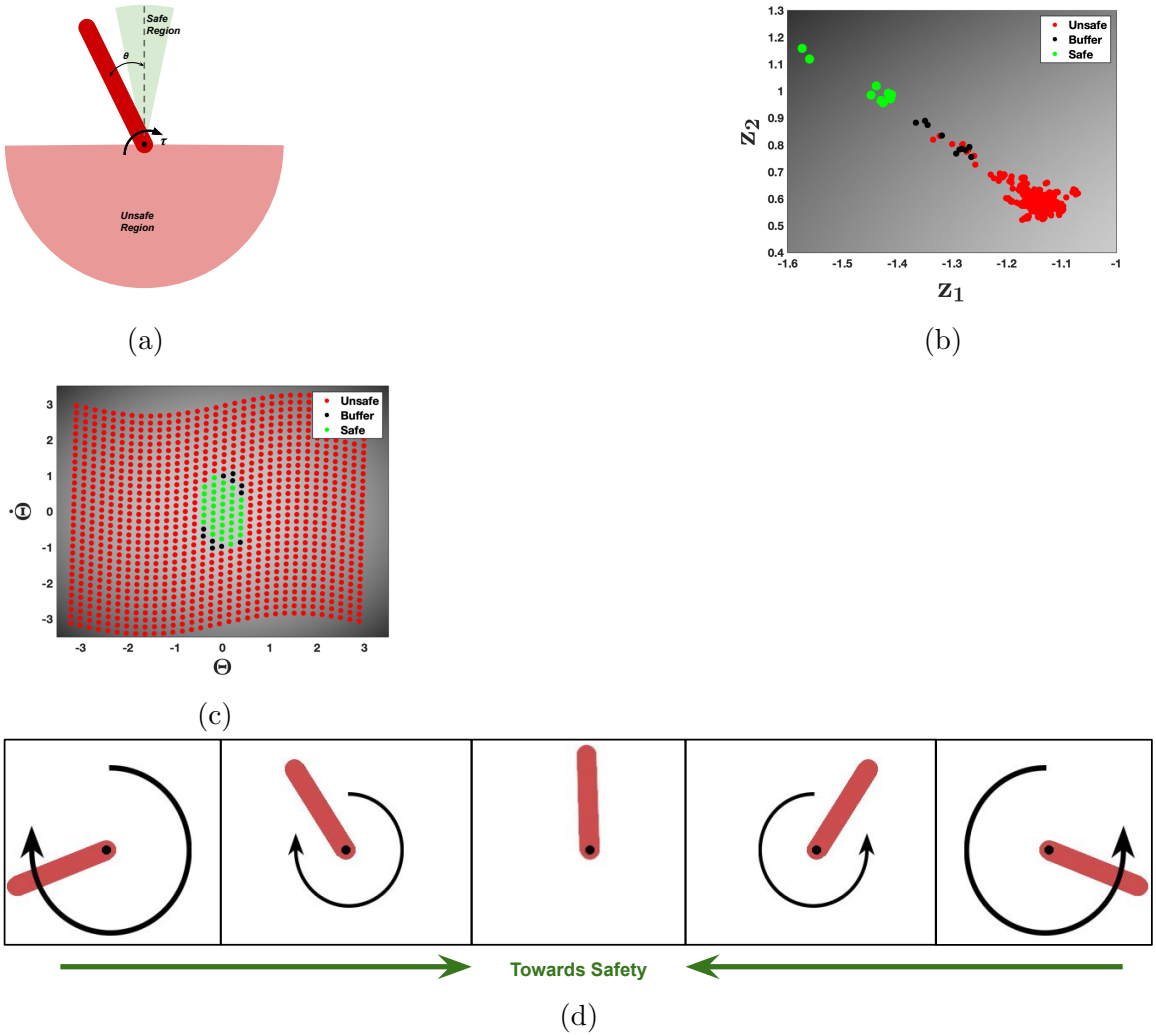


Figure 10.2: This figure demonstrates the results of our framework applied to OpenAI Gym's Pendulum environment. **Fig.** 10.2a illustrates the system diagram, where the light green area represents the safe region (used for sampling safe states, $\mathcal{S}$), and the light red area denotes the unsafe region (used for sampling unsafe states, $\mathcal{U}$). **Fig.** 10.2b shows the representation of safe, unsafe and buffer states. The figure shows that the safe and unsafe regions are well segregated. **Fig.** 10.2c visualizes the safe, unsafe, and buffer points in the $\Theta - \dot{\Theta}$ plane. We observe that the boundary of the trained barrier function (black dots) successfully separates the unsafe and safe regions. **Fig.** 10.2d shows the pendulum's trajectory generated by the learned policy π_θ, where the size of the arrows indicates the magnitude of the control actions produced by the policy.

10.2a. The discrete-time dynamics governing this system are given by:

$$\begin{bmatrix} \Theta_{t+1} \\ \dot{\Theta}_{t+1} \end{bmatrix} = \begin{bmatrix} \Theta_t \\ \dot{\Theta}_t \end{bmatrix} + \left(\begin{bmatrix} \dot{\Theta}_t \\ \frac{g}{l}\sin(\Theta_t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u_t \right) dt, \mathcal{O}_t = Img(x_t, x_{t-1}), \quad (10.13)$$

where m and l represent the mass and length of the pendulum, respectively, and Img represents the stack of RGB frames at two consecutive time steps. The datasets, $\mathcal{S}$,$\mathcal{U}$, and $\mathcal{D}$, are sampled as follows:

$$\begin{aligned} \mathcal{S} &= \{\mathcal{O}_t | x_t \in \left[-\frac{\pi}{12}, \frac{\pi}{12}\right] \times [-0.25, 0.25]\} \\ \mathcal{U} &= \{\mathcal{O}_t | x_t \in [-\pi, \pi] \times [-3.5, 3.5] \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [-1.5, 1.5]\} \\ \mathcal{D} &= \{\mathcal{O}_t | x_t \in [-\pi, \pi] \times [-3.5, 3.5]\}. \end{aligned} \quad (10.14)$$

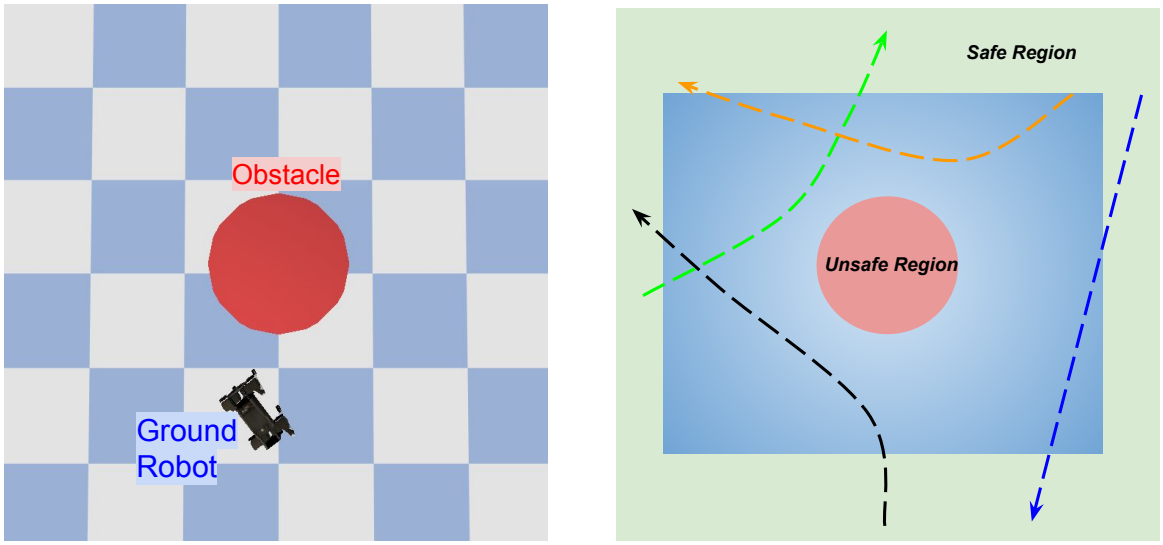
The barrier network used in this case is Lipschitz bounded with $L_B = 2$, and the latent dimension size is set to 2. A user-defined policy, π_{user}, is set to 0 to enable the training of a robust controller.

Visualizations of the trained barrier function are presented in both latent space with sample points (Fig. 10.2b) and in $\Theta - \dot{\Theta}$ plane (Fig. 10.2c). These visualizations demonstrate the successful separation of the safe region from the unsafe region. As shown in Fig. 10.2d, the trajectories generated by the learned policy π_θ start in the unsafe region and successfully reach the safe region, validating our approach.

Figure 10.4 shows that our method guarantees safety, unlike Imitation Learning and PPO-Lagrangian, which achieve lower rates. Training without formal verification still outperforms the baselines but does not guarantee safety, highlighting the importance of formal assurances.

10.5.2 Obstacle Avoidance on Autonomous Ground Vehicle

In our next case study, we analyze an obstacle avoidance problem for an autonomous ground robot, modeled here as a reduced-order representation of a quadruped robot [156, 155, 110]. The state vector is defined as $[x_t, y_t, \Theta_t]^T \in [-2, 2]^2 \times [-\pi, \pi]$, where (x_t, y_t) denotes the robot's position and Θ_t represents its orientation. The robot moves at a constant speed $v = 1$, while the visuomotor observation $\mathcal{O}_t$ consists of a stack of two 128×128 RGB frames. The control input u corresponds to the yaw rate ω.



(a) (b)

Figure 10.3: Presents the results on the velocity-driven quadruped model on PyBullet. Figure 10.3b depicts the x–y plane, where the light green and red regions indicate safe ($\mathcal{S}$) and unsafe ($\mathcal{U}$) states used for sampling. Trajectories starting in the safe set follow the policy π_θ synthesized by our framework.

It follows the discrete-time dynamics given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \Theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \Theta_t \end{bmatrix} + \left(\begin{bmatrix} v\cos\Theta_t \\ v\sin\Theta_t \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_t \right) dt, \mathcal{O}_t = Img(x_t, x_{t-1}), \quad (10.15)$$

where Img represents the stack of RGB frames at two consecutive time steps. Fig 10.3a shows a sample input image at a particular time instant. The datasets, $\mathcal{S}$,$\mathcal{U}$,$\mathcal{D}$, are sampled as follows:

$$\begin{aligned} \mathcal{S} &= \{\mathcal{O}_t | x_t \in [-2,2]^2 \times [-\pi,\pi] \backslash [-1.5,1.5]^2 \times [-\pi,\pi]\} \\ \mathcal{U} &= \{\mathcal{O}_t | x_t \in [-0.7,0.7]^2 \times [-\pi,\pi]\} \\ \mathcal{D} &= \{\mathcal{O}_t | x_t \in [-2,2]^2 \times [-\pi,\pi]\}. \end{aligned} \quad (10.16)$$

The barrier network used in this case is Lipschitz with a constant $L_B = 1.5$, and the latent dimension size is set to 4.

We assess the performance of the proposed safe policy within the Quadruped environment using the PyBullet simulation framework [39]. The quadruped robot is equipped with a low-level controller, such as Convex MPC, which facilitates precise velocity tracking along the x, y, and yaw axes. Figure 10.3b illustrates the trajectories generated by the learned policy π_θ, which successfully avoids the obstacle and reaches the safe region, thereby validating our approach.

Figure 10.4 shows a trend consistent with the inverted pendulum case: our method guarantees safety, while Imitation Learning and PPO-Lagrangian achieve lower rates. Without formal verification, performance declines, highlighting the need for formal assurances.



Figure 10.4: Empirical safety comparison across methods. The verified SaLaD variant achieves the strongest observed safety performance in the tested settings, while GCIL and PPO-Lagrangian do not enforce the latent CBC condition. The unverified ablation illustrates the empirical benefit of integrating the proposed certificate during training.

10.6 Chapter Summary

We introduced a semi-supervised framework that synthesizes a Control Barrier Certificate (CBC) and a policy from visuomotor observations. Under the stated cover and Lipschitz assumptions, the sampled CBC inequalities extend over the covered latent region; physical safety additionally depends on latent-dynamics fidelity and preservation of safety-relevant information by the encoder. The derived consistency condition bounds the learned-dynamics residual on the training samples, while a region-wide guarantee would require an error bound throughout the covered region. Experiments on an inverted pendulum and an autonomous mobile robot provide empirical evidence that the framework improves safety relative to the evaluated baselines. Future work will extend the framework to more complex high-dimensional systems, derive regional perception and dynamics-error bounds, and validate performance on physical hardware.

Bridge to the Discussion. This chapter completes the technical arc of the dissertation. Starting from classical analytical CBFs for ground and aerial robots (Chapter 6), we progressively developed learning-based safety certificates with stochastic (Chapter 3) and conformal (Chapter 4) verification, integrated safety with performance via offline RL (Chapter 7) and physics-informed optimal control (Chapters 8–9), and finally extended formal safety to high-dimensional perception (this chapter). The next chapter synthesizes these contributions, discusses their interconnections and limitations, and outlines promising directions for future research.

Chapter 11

Discussion and Future Directions

This dissertation set out to answer a question posed in the opening chapter: *how can safety and optimality certificates be synthesized directly from data while retaining formal guarantees under uncertainty?* Across eight technical chapters, we developed a progression of methods that chart a path from verified safety certificates, through planning-aware and performance-co-optimized control, to safe decision-making from raw sensory observations.

11.1 Summary of Contributions

The thesis began with the challenge of constructing safety certificates without hand-engineering. In Part I, we showed that neural networks can learn Control Barrier Functions directly from data: first for both deterministic and stochastic systems, where Lipschitz-based LMI constraints embedded *during* training yield deterministic guarantees over the entire continuous state space (Chapter 3), then for deterministic systems, where conformal prediction provides distribution-free, finite-sample certificates that are significantly less conservative (Chapter 4), and finally through a Verification-In-The-Loop (VITL) synthesis framework that replaces dense sampling with NLP-based verifiers and extends stochastic certificates to ReLU-activated networks via Tanaka's formula, achieving substantially larger certified safe sets (Chapter 5). Together, these three frameworks establish a complete pipeline for learning and verifying neural safety certificates using complementary verification paradigms, but enforce safety in a myopic, single-step sense.

Part II addressed this limitation by embedding temporal look-ahead into the safety framework. Collision Cone CBFs (Chapter 6) combine geometric planning primitives

with barrier functions to provide velocity-level foresight for collision avoidance, validated on ground and aerial robots in both simulation and hardware. Moving from this task-specific construction to a general-purpose one, V-OCBF (Chapter 7) unifies Hamilton–Jacobi reachability, offline reinforcement learning, and CBFs through a common value-function lens, producing actuation-aware barrier certificates that account for long-horizon constraint satisfaction without requiring known dynamics or online rollouts.

With safety certification in hand, Part III turned to the joint optimization of safety and performance. By reformulating the state-constrained optimal control problem via an epigraph transformation and solving the resulting HJI variational inequality with physics-informed neural networks, we obtained value functions and controllers with distribution-relative safety and performance certificates from conformal calibration (Chapter 8). A decentralized extension with reachability-based neighbor selection scaled the physics-informed policy architecture to multi-agent settings with up to 256 agents (Chapter 9); its safety and scalability evidence is empirical rather than a conformal certificate.

Finally, Part IV closed the perception gap. The SaLaD framework (Chapter 10) lifts barrier certificate synthesis from full-state feedback to learned latent representations of high-dimensional visual inputs, enabling safe visuomotor control directly from camera images without explicit state estimation.

Two methodological threads run across the entire dissertation and bind the chapters into a coherent framework rather than a collection of independent results. *Conformal prediction* serves as a unifying verification tool, introduced for barrier verification in Chapter 4 and reused for safety and performance certification in Chapter 8. *Physics-informed learning* provides a scalable PDE solver, introduced in Chapter 8 for single-agent optimal control and directly extended to the multi-agent setting in Chapter 9. A third thread, *verification-integrated training*, connects the Lipschitz LMI constraints of Chapter 3 with the VITL counterexample-guided synthesis of Chapter 5, showing how formal verification can be woven into the learning process itself. These shared threads ensure that each contribution both stands alone and strengthens the whole.

11.2 Scope of Guarantees and Assumptions Across Chapters

Because the technical chapters deliberately employ different verification paradigms, we summarize here—in one place—the type of guarantee delivered by each chapter,

what exactly is certified, and the assumptions under which each claim holds. We emphasize that these guarantees form a *spectrum of strength* rather than a single uniform claim: deterministic forward invariance (the strongest), distribution-free probabilistic coverage, and coverage-relative assurances are successively weaker notions, traded for scalability and conservatism along the way. In particular, the conformal certificates of Chapters 4 and 8 bound the *probability* of encountering a violated condition over the sampled state distribution; they are intentionally not set-invariance guarantees, which only the Lipschitz-based and collision-cone constructions provide.

We hope this consolidated view clarifies precisely which claims are invariant-certificates, which are probabilistic, and which are conditional on learned-component fidelity—and thereby makes explicit both the strength and the boundary of each contribution.

11.3 Future Challenges and Opportunities

The methods developed in this dissertation address several fundamental questions in safe autonomy, but they also expose a number of open problems whose resolution would substantially advance the field. Below, we outline five research directions that we believe are both technically promising and practically important. Each direction is motivated by a specific limitation of the current framework and sketches a concrete path forward.

11.3.1 From Static Certificates to Adaptive Safety

A common assumption across this dissertation is that the barrier function and its associated safe set are fixed after training. In practice, environments change: new obstacles appear, terrain shifts, or operating conditions evolve. Two complementary strategies could address this.

First, combining barrier certificates with learned *world models* [63] could enable *predictive safety*: anticipating future unsafe states through model-based rollouts and adjusting the barrier function accordingly. The physics-informed framework of Chapter 8 provides a natural starting point, since the HJI variational inequality already encodes forward-looking safety; replacing the known dynamics with a learned world model would extend this capability to model-free settings, provided the resulting model error is incorporated into calibration rather than assumed away. For highway

Chapter	Guarantee Type	What Is Certified	Key Assumptions
3 (Lipschitz NCBF)	Formal, deterministic	Validity of the CBF conditions at *every* x in an $\bar{\epsilon}$-coverable state space, i.e., forward invariance of $\widehat{\mathcal{C}}$	Global Lipschitz bounds on B_θ, ∇B_θ (enforced by LMIs during training); known (f, g); compact $\mathcal{X}$
4 (CP-NCBF)	Distribution-free, probabilistic	$\mathbb{P}\big(\max_k q_k(x) \le \hat{q}\big) \ge 1-\epsilon$ with confidence $1-\beta$ over $\mathcal{X}$; margin $\psi = -\hat{q}$ fed back into training	Exchangeability of calibration samples; known (f, g); residual violation probability $\epsilon > 0$ remains by design
5 (VITL SNCBF)	Worst-case probability (stochastic)	Region-wise satisfaction of the stochastic CBF condition (incl. Itô correction) via the stated NLP/ReLU-enumeration procedure, subject to solver tolerances and completeness	Twice-differentiable activations (smooth case) or ReLU regions (exact enumeration); Lipschitz bounds incl. Hessian term; diagonal diffusion
6 (C3BF)	Formal, deterministic	Forward invariance of a collision-cone-derived safe set for the single-obstacle setting; relative-degree-one handling of velocity-level constraints	Constant-velocity obstacles; circular over-approximation of obstacle geometry; model knowledge; small-slip-angle approximation (bicycle); simultaneous multi-obstacle QP feasibility is not guaranteed
7 (V-OCBF)	Conditional (hard constraints, learned ingredients)	Barrier conditions enforced as hard QP constraints at sampled states using finite-difference condition; safe-set expansion beyond demonstrations	Fidelity of learned dynamics and CBVF surrogate; dataset coverage; error bounds not yet propagated (see Chapter summary)
8 (PINN SOC)	Probabilistic (conformal) for safety *and* performance	Distribution-relative violation levels δ_s, δ_p for safety and performance of the value-based policy; numerical HJI-VI approximation	Assumptions $\mathcal{A}_1'$–$\mathcal{A}_4'$; known dynamics; collocation coverage of augmented state space; exchangeability for conformal stage
9 (MAD-PINN)	Empirical, distribution-relative evaluation	Safety rate, fully safe scenarios, and cumulative cost for the tested agent distributions; deployment of unchanged policy weights up to 256 agents	Cooperative homogeneous agents; fixed local observation structure; reachability-based neighbor selection; tested densities, horizons, and initial-condition distributions
10 (SaLaD)	Coverage-relative, conditional	Sampled CBC inequalities extended over the $\bar{\epsilon}$-covered *latent* region; learned-dynamics consistency bounded on training samples	Lipschitz residuals on the covered region; regional latent-dynamics fidelity; encoder retains safety-relevant information; deployment scenes represented by the training distribution

Table 11.1: Summary of the guarantee type, certified or evaluated statement, and key assumptions for each technical chapter. The rows distinguish deterministic invariance, distribution-free probabilistic coverage, conditional learned-component assurances, and empirical evaluation.

and changing-terrain applications, this extension could combine road-structure priors with terrain-conditioned dynamics and barrier models, while robust or stochastic margins account for uncertainty in friction, slope, and obstacle motion.

Second, combining online conformal prediction with streaming deployment data could yield *adaptive barrier functions* whose safety margins tighten or relax in response to observed distribution shifts. Rather than committing to a fixed confidence set at training time, the system would continuously update its uncertainty estimate, bridging the offline training of Chapters 3–4 with real-time deployment requirements. This is closely connected to the broader challenge of safe online learning: developing principled strategies for collecting safety-critical data (through adversarial sampling, curiosity-driven exploration constrained by safety, or targeted augmentation near the safe-set boundary) could significantly improve both the efficiency and the reliability of learned certificates throughout their operational lifetime.

11.3.2 Safe Autonomy in Strategic and Human-Centered Environments

The multi-agent framework of Chapter 9 assumes cooperative, homogeneous agents that share a joint objective. Real-world autonomy is rarely so benign. Autonomous vehicles at unsignalized intersections must contend with strategic drivers whose intentions are unknown; service robots in hospitals must anticipate unpredictable human motion; drone teams may comprise heterogeneous platforms with different dynamics, sensing modalities, and communication constraints. These settings demand safety frameworks that reason about *strategic uncertainty* and *latent intent*.

One concrete path is to extend the physics-informed safety framework from HJB to *Hamilton–Jacobi–Isaacs* (HJI) equations, which encode two-player differential games and naturally capture adversarial or rational-but-self-interested co-players. Training physics-informed networks to solve the resulting differential game would enable safe coordination even when agents cannot be trusted to cooperate. Conformal prediction (Chapter 4) could provide finite-sample safety certificates under worst-case opponent behavior.

A closely related challenge arises when the "opponent" is a human. Safety then depends on quantities that are not directly observable (intent, attention, and future motion), and barrier certificates must be defined over *joint physical–information state spaces* that augment the robot's configuration with its belief about the human's latent state. The stochastic CBF framework of Chapter 3 offers a viable foundation: the SDE formulation can accommodate belief dynamics as additional stochastic states, and

the Lipschitz-based training-time verification, which jointly enforces LMI constraints and the validity condition $L_{\max}\bar{\epsilon} + \psi^* \leq 0$ during learning, extends naturally to the augmented state space. Coupling this with online conformal updates as the robot gathers observations of human behavior could yield barrier functions that actively trade information gain against safety, slowing cautiously when intent is ambiguous and acting decisively once the belief concentrates. We view this integration of game-theoretic reasoning, belief-space planning, and verified safety certificates as one of the most important open frontiers in trustworthy autonomy.

11.3.3 Scalable Verified Learning for Neural Certificates

This dissertation introduced three complementary verification paradigms: Lipschitz-based LMI constraints embedded during training (Chapter 3), conformal prediction applied to the trained network (Chapter 4), and verification-in-the-loop synthesis with NLP-based verifiers (Chapter 5). Each has distinct strengths and limitations. Lipschitz-based verification provides deterministic guarantees over the covered continuous state-space domain but introduces conservatism through global bounds and is currently restricted to single-hidden-layer architectures (Theorem 3.4.2). Conformal prediction is distribution-free and architecture-agnostic but offers probabilistic rather than deterministic guarantees. VITL produces larger certified safe sets on the tested architectures but requires solving NLPs or enumerating activation regions at each verification step; the current ReLU formulation and experiments do not establish general deep-network scalability.

Bridging this gap requires advances on two fronts. On the *architecture* side, developing compositional or layer-wise LMI certificates for deeper networks (for instance, by propagating quadratic constraints through successive layers or using Lipschitz-bounded architectures such as Cayley layers and orthogonal convolutions) would preserve the *training-integrated* nature of Lipschitz verification while enabling the representational power needed for high-dimensional systems. The VITL framework of Chapter 5 takes a complementary approach by moving verification outside the training loop; combining its targeted counterexample discovery with LMI-guided training could reduce verification cost. A further route for large multi-agent systems is compositional verification: certify pairwise or local barriers over fixed-size neighborhoods, then establish conditions under which their conjunction preserves safety and QP feasibility for the full decentralized system. On the *theory* side, providing formal convergence guarantees for the joint (θ, ψ, Λ) optimization would strengthen training-time verification and clarify how network depth, dataset size, and sampling density determine certificate tightness. A multi-stage framework in which LMI constraints

guide training, VITL verifiers refine the barrier, and conformal prediction provides a distribution-free deployment wrapper is a natural next step.

11.3.4 Closing the Sim-to-Real Gap for Learned Safety

Several methods in this dissertation (Chapters 6, 4, 8, and 9) assume known dynamics, and the learning-based methods (Chapters 3–10) were validated primarily in simulation. Deploying neural safety certificates on physical robots therefore remains an essential and largely open challenge.

The C3BF hardware experiments of Chapter 6 provide a validation template: closed-form barrier functions with known dynamics were evaluated on unicycle and quadrotor platforms using instrumented localization and offboard components. Extending this to learned certificates requires addressing model mismatch, sensor noise, actuator delays, onboard computational limits, and the distributional gap between simulation and the real world. Domain randomization and system identification can reduce model error, but the more fundamental question is how model inaccuracy propagates through the safety certificate. Robust conformal prediction under distribution shift—a natural extension of the conformal framework developed in Chapter 4—could provide deployment-time safety bounds that account for the sim-to-real gap without requiring exact knowledge of the shift. Combining this with adaptive online updates (as discussed in Section 11.3.1) would yield a complete pipeline: train in simulation with Lipschitz or conformal guarantees, deploy with robust conformal margins, and tighten those margins online as real-world data accumulates. Fully onboard, perception-coupled multi-agent validation without motion-capture infrastructure remains an important deployment milestone.

11.3.5 Input Constraints and Certified Feasibility

Although several formulations in this dissertation specify bounded input sets, most chapter-level certificate arguments do not establish recursive feasibility of the deployed optimization under those bounds. Enforcing joint state–input constraints is therefore an important extension: barrier conditions can become infeasible when the safety filter demands controls outside $\mathcal{U}$, and recursive feasibility of the resulting QPs must then be established rather than assumed. Promising routes include computing control-invariant sets under actuation limits via Hamilton–Jacobi reachability with bounded input sets—whose value functions already encode input constraints by construction and which slot naturally into the physics-informed pipeline of Chapter 8—as well as penalty-based soft enforcement accompanied by feasibility certificates, and extending

the epigraph reformulation to co-optimize over admissible input sets so that certified policies are feasible by construction.

11.3.6 Verified Safety for Foundation Models in Robotics

Large-scale foundation models for robotics [23] offer unprecedented generalization across tasks and environments, a capability that none of the methods in this dissertation currently possess. However, these models lack the formal safety guarantees demanded by safety-critical deployment. Integrating barrier-certificate-based safety filters with foundation-model policies could yield *safe foundation policies* that inherit the generalization of large models while maintaining provable safety.

The visuomotor framework of Chapter 10 provides a natural pathway: just as the SaLaD framework learns a barrier certificate in the latent space of a dynamics model, one could learn a barrier certificate in the representation space of a vision–language–action (VLA) foundation model and use it as a real-time safety filter on the model's outputs. The key technical challenge is that the representation space of a foundation model is not designed for safety (it optimizes for task generalization), and the safety certificate must therefore be *robust* to representational drift as the foundation model is fine-tuned or prompted for new tasks. Conformal prediction offers a promising tool for providing task-conditional safety guarantees in this setting, and the training-time Lipschitz verification of Chapter 3 could regularize the barrier network to remain certifiably safe across representational perturbations. We view this integration as a high-impact direction that connects the theoretical contributions of this thesis with the most practically significant trend in modern robotics.

11.4 Closing Remarks

This dissertation opened with a question: *how can safety and optimality certificates be synthesized directly from data while retaining formal guarantees under uncertainty?* The eight technical chapters that followed developed one answer: not a single monolithic framework, but a progression of methods that systematically integrate the mathematical structure of control theory with the expressiveness of machine learning. From collision cone CBFs to neural barriers, from Lipschitz verification to conformal prediction, from safety-only certificates to safety–performance co-optimization, and from full-state feedback to raw visual perception, each contribution addressed a specific limitation of the preceding one, tracing an arc from verified safety to trustworthy autonomy.

The future directions outlined above suggest that this arc is far from complete. As autonomous systems move from controlled laboratories into shared human spaces (hospitals, warehouses, roads, homes), the need for methods that are simultaneously safe, adaptive, and performant will only intensify. We hope that the ideas and tools developed in this dissertation serve as useful building blocks toward that goal, and that the open problems identified here inspire further progress at the intersection of learning, control, and verification.

Appendices

Appendix A

Appendix of Chapter 6

A.1 Proofs

A.1.1 Proof of Theorem 6.4.3

Theorem 6.4.3 (C3BF Validity for Unicycle Model) *Given the acceleration controlled unicycle model (6.1) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = 0$), the proposed CBF candidate (6.8) with p_{rel}, v_{rel} defined by (6.9), (6.10) is a valid CBF defined for the set $\mathcal{D}$.*

Proof. Taking the derivative of (6.8) yields

$$\begin{aligned}\dot{B} = &< \dot{p}_{rel}, v_{rel} > + < p_{rel}, \dot{v}_{rel} > \\ &+ < v_{rel}, \dot{v}_{rel} > \frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} \\ &+ < p_{rel}, \dot{p}_{rel} > \frac{\|v_{rel}\|}{\sqrt{\|p_{rel}\|^2 - r^2}}.\end{aligned} \tag{A.1}$$

Further $\dot{p}_{rel} = v_{rel}$ and

$$\dot{v}_{rel} = \begin{bmatrix} -a\cos\theta + v(\sin\theta)\omega + l(\cos\theta)\omega^2 + l(\sin\theta)\alpha \\ -a\sin\theta - v(\cos\theta)\omega + l(\sin\theta)\omega^2 - l(\cos\theta)\alpha \end{bmatrix}.$$

Given $\dot{v}_{rel}$ and $\dot{B}$, we have the following expression for $\mathcal{L}_g B$:

$$\mathcal{L}_g B = \begin{bmatrix} < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \begin{bmatrix} -\cos\theta \\ -\sin\theta \end{bmatrix} > \\ < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \begin{bmatrix} l\sin\theta \\ -l\cos\theta \end{bmatrix} > \end{bmatrix}^T, \tag{A.2}$$

It can be verified that for $\mathcal{L}_g B$ to be zero, we can have the following scenarios:

- $p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} = 0$, which is not possible. Firstly, $p_{rel} = 0$ indicates that the vehicle is already inside the obstacle. Secondly, if the above equation were to be true for a non-zero p_{rel}, then $v_{rel}/\|v_{rel}\| = -p_{rel}/\sqrt{\|p_{rel}\|^2 - r^2}$. This is also not possible as the magnitude of LHS is 1, while that of RHS is > 1.
- $p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}$ is perpendicular to both $\begin{bmatrix} -\cos\theta \\ -\sin\theta \end{bmatrix}$ and $\begin{bmatrix} l\sin\theta \\ -l\cos\theta \end{bmatrix}$, which is also not possible.

This implies that $\mathcal{L}_g B$ is always a non-zero matrix, implying that B is a valid CBF. □

Remark A.1.1. *Since $\mathcal{L}_g B \neq 0$, we can infer from [191, Theorem 8] that the resulting QP given by (2.11) is Lipschitz continuous. Hence, we can construct CBF-QPs with the proposed CBF (6.8) for the unicycle model and guarantee collision avoidance. In addition, if $B(x(0)) < 0$, then we can construct a class $\mathcal{K}$ function κ in such a way that the magnitude of B exponentially decreases over time, thereby minimizing the violation. We will demonstrate these scenarios in Section 6.5.*

A.1.2 Proof of Theorem 6.4.4

Theorem 6.4.4 (C3BF Validity for Bicycle Model) *Given the bicycle model (6.2) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = 0$), the proposed candidate CBF (6.8) with p_{rel}, v_{rel} defined by (6.11), (6.12) is a valid CBF defined for the set $\mathcal{C}$.*

Proof. We need to show that $\mathcal{L}_g B = 0 \implies \dot{B} + \kappa(B) \geq 0$. The derivative of B

(6.8) yields: Taking the derivative of (6.8) yields

$$\dot{B} = \frac{\partial B}{\partial t} + \frac{\partial B}{\partial x} \cdot \dot{x} = < \dot{p}_{rel}, v_{rel} > + < p_{rel}, \dot{v}_{rel} > \\ + < v_{rel}, \dot{v}_{rel} > \frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} \\ + < p_{rel}, \dot{p}_{rel} > \frac{\|v_{rel}\|}{\sqrt{\|p_{rel}\|^2 - r^2}}. \tag{A.3}$$

Further using (6.2): $\dot{p}_{rel} = v_{rel} + \beta[v \sin\theta, -v\cos\theta]^T$ and

$$\dot{v}_{rel} = \begin{bmatrix} -\cos\theta & v\sin\theta \\ -\sin\theta & -v\cos\theta \end{bmatrix} \begin{bmatrix} a \\ \frac{v}{l_r}\beta \end{bmatrix}. \tag{A.4}$$

When $\mathcal{L}_g B = 0$, we have

$$\dot{B} + \kappa(B) =< v_{rel}, v_{rel} > + \frac{< p_{rel}, v_{rel} > \|v_{rel}\|}{\sqrt{\|p_{rel}\|^2 - r^2}} + \kappa(B).$$

Rewriting the above equation yields

$$\frac{\|v_{rel}\|}{\sqrt{\|p_{rel}\|^2 - r^2}} \left(B + \frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} \kappa(B) \right). \tag{A.5}$$

Since $\sqrt{\|p_{rel}\|^2 - r^2}$ and $\|v_{rel}\|$ are positive quantities, the entire quantity above is ≥ 0 for all $x \in \mathcal{C}$. This completes the proof. □

Remark A.1.2. *Theorem 6.4.4 is different from Theorem 6.4.3 as the CBF inuquality is satisfied in the set $\mathcal{C}$ and not in $\mathcal{D}$. In other words, forward invariance of $\mathcal{C}$ can be guaranteed, but not asymptotic convergence of $\mathcal{C}$. However modifications of the control formulation is possible to extend the result for $\mathcal{D}$, which will a subject of future work.*

A.1.3 Proof of Theorem 6.4.5

Theorem 6.4.5 (C3BF Validity for Quadrotor Model) *Given the quadrotor model (6.3) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = \ddot{c}_z = 0$), the proposed CBF candidate (6.8) with p_{rel}, v_{rel} defined by (6.13), (6.14) is a valid CBF defined for the set $\mathcal{D}$.*

Proof. Taking the derivative of (6.8) yields

$$\begin{aligned}\dot{B} = \frac{\partial B}{\partial t} + \frac{\partial B}{\partial x} \cdot \dot{x} = &< \dot{p}_{rel}, v_{rel} > + < p_{rel}, \dot{v}_{rel} > \\ &+ < v_{rel}, \dot{v}_{rel} > \frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} \\ &+ < p_{rel}, \dot{p}_{rel} > \frac{\|v_{rel}\|}{\sqrt{\|p_{rel}\|^2 - r^2}}. \end{aligned} \tag{A.6}$$

By substituting for $\dot{v}_{rel}$ (which contains the input) in $\dot{B}$ (A.6), we have the following expression for $\mathcal{L}_g B$:

$$\mathcal{L}_g B = \begin{bmatrix} < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \mathbf{R} \begin{bmatrix} 0 \\ \frac{-Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix} > \\ < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \mathbf{R} \begin{bmatrix} \frac{Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix} > \\ < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \mathbf{R} \begin{bmatrix} 0 \\ \frac{Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix} > \\ < p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}, \mathbf{R} \begin{bmatrix} \frac{-Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix} > \end{bmatrix}^T, \tag{A.7}$$

It can be verified that for $\mathcal{L}_g B$ to be zero, we can have the following scenarios:

- $p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|} = 0$, which is not possible. Firstly, $p_{rel} = 0$ indicates that the vehicle is already inside the obstacle. Secondly, if the above equation were to be true for a non-zero p_{rel}, then $v_{rel}/\|v_{rel}\| = -p_{rel}/\sqrt{\|p_{rel}\|^2 - r^2}$. This is also not possible as the magnitude of LHS is 1, while that of RHS is > 1.
- $p_{rel} + v_{rel}\frac{\sqrt{\|p_{rel}\|^2 - r^2}}{\|v_{rel}\|}$ is perpendicular to all $\mathbf{R} \begin{bmatrix} 0 \\ \frac{-Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix}$, $\mathbf{R} \begin{bmatrix} \frac{Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix}$, $\mathbf{R} \begin{bmatrix} 0 \\ \frac{Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix}$ and $\mathbf{R} \begin{bmatrix} \frac{-Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix}$, which is also not possible. (Because three of these vectors form basis vectors for $\mathbb{R}^3$)

This implies that $\mathcal{L}_g B$ is always a non-zero matrix, implying that B is a valid CBF. □

A.1.4 Proof of Theorem 6.4.6

Theorem 6.4.6 (C3BF Validity for Projected Quadrotor Model) *Given the quadrotor model (6.3) with the assumptions that the obstacles are moving at a constant velocity ($\ddot{c}_x = \ddot{c}_y = \ddot{c}_z = 0$), in a direction perpendicular to the cylindrical axis with no rotation, the proposed CBF candidate (6.22) with p_{rel}, v_{rel} defined by (6.15), (6.17) is a valid CBF defined for the set $\mathcal{D}$.*

Proof. We have the following derivative of B_{proj}:

$$\dot{B}_{proj} = \frac{\partial B_{proj}}{\partial t} + \frac{\partial B_{proj}}{\partial x} \cdot \dot{x} \tag{A.8}$$

$$< (\dot{p}_{rel})_{proj}, (v_{rel})_{proj} > + < (p_{rel})_{proj}, (\dot{v}_{rel})_{proj} > + < (v_{rel})_{proj}, (\dot{v}_{rel})_{proj} > \frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(v_{rel})_{proj}\|} + < (p_{rel})_{proj}, (\dot{p}_{rel})_{proj} > \frac{\|(v_{rel})_{proj}\|}{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}. \tag{A.9}$$

$\dot{v}_{rel}$ (which contains the input) from (6.19), equations (6.20), (6.21) and $\dot{B}_{proj}$ (A.9), we have the following expression for $\mathcal{L}_g B_{proj}$:
$\mathcal{L}_g B_{proj} =$

$$\begin{bmatrix} < (p_{rel})_{proj} + (v_{rel})_{proj} \frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(v_{rel})_{proj}\|}, \mathbf{R} \begin{bmatrix} 0 \\ \frac{-Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix} > \\ < (p_{rel})_{proj} + (v_{rel})_{proj} \frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(v_{rel})_{proj}\|}, \mathbf{R} \begin{bmatrix} \frac{Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix} > \\ < (p_{rel})_{proj} + (v_{rel})_{proj} \frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(v_{rel})_{proj}\|}, \mathbf{R} \begin{bmatrix} 0 \\ \frac{Ll}{I_{xx}} \\ \frac{1}{m} \end{bmatrix} > \\ < (p_{rel})_{proj} + (v_{rel})_{proj} \frac{\sqrt{\|(p_{rel})_{proj}\|^2 - r^2}}{\|(v_{rel})_{proj}\|}, \mathbf{R} \begin{bmatrix} \frac{-Ll}{I_{yy}} \\ 0 \\ \frac{1}{m} \end{bmatrix} > \end{bmatrix}^T, \tag{A.10}$$

Using the same arguments we gave in the proof of the theorem. 6.4.5, we can infer that $\mathcal{L}_g B_{proj}$ cannot be zero. This implies that B_{proj} is a valid CBF. □

Remark A.1.3. *Based on Theorems (6.4.5) and (6.4.6), where $\mathcal{L}_g B \neq 0$, we can utilize the conclusion from [191, Theorem 8] to deduce that the control inputs obtained from the resulting CBF-QP (2.11) are Lipschitz continuous. As a result,*

solutions guarantee forward invariance of the safe set generated by the proposed C3BF candidates.

Appendix B

Appendix of Chapter 3

B.1 Proofs

B.1.1 Proof of Theorem 3.3.1

Theorem 3.3.1 (Validity Condition for Neural CBF) *Consider a continuous-time control-affine system (2.1), and initial safe and unsafe sets $\mathcal{X}_s \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$, respectively. Let $\tilde{B}(x \mid \theta)$ be the neural network-based CBF with trainable parameters θ. For the SOP (3.4) constructed by utilizing N samples, let ψ^* be the optimal value, with the assumption that functions $q_k(x), k \in \{1,2,3\}$ in equation (3.3) are Lipschitz continuous. Then $\tilde{B}(x \mid \theta)$ is a valid CBF, i.e., it satisfies conditions (3.1), if $L_{\max}\bar{\epsilon} + \psi^* \leq 0$, where $L_{\max}$ is the maximum of the Lipschitz constants of $q_k(x), k \in \{1,2,3\}$.*

Proof. For any x and any $k \in \{1,2,3\}$, we know that:

$$\begin{aligned} q_k(x) &= q_k(x) - q_k(x_i) + q_k(x_i) \\ &\leq L_k \|x - x_i\| + \psi^* \\ &\leq L_k\bar{\epsilon} + \psi^* \leq L_{\max}\bar{\epsilon} + \psi^* \leq 0. \end{aligned}$$

Hence, if $q_k(x), k \in \{1,2,3\}$ satisfies condition (3.5), then the $\tilde{B}(x \mid \theta)$ is a valid CBF, satisfying conditions (3.1). □

B.1.2 Proof of Theorem 3.4.2

Theorem 3.4.2 (Effective Weights for Lipschitz Certification) *Consider a 1-layered feedforward neural network f_θ, with output dimension 1×1, where θ represents the trainable weight parameters. Let y denote the output, x the input, θ_i, $i \in \{0, 1\}$, the weight parameters of each layer with $\theta = \{\theta_0, \theta_1\}$ and ϕ the activation. The certificate for L-Lipschitz continuity of $\frac{\partial y}{\partial x}$ is given by $M_{\hat{\phi}}(\hat{\theta}, \Lambda) \succeq 0$, where $\hat{\phi} = \phi'$ and $\hat{\theta} = (\theta_0, \hat{\theta}_1)$ with $\hat{\theta}_1 = \theta_0^T \mathsf{diag}(\theta_1)$. Similarly, the certificate for L-Lipschitz continuity of $\mathsf{tr}(\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma)$ is $M_{\bar{\phi}}(\bar{\theta}, \Lambda) \succeq 0$, where $\bar{\phi} = \phi''$ and $\bar{\theta} = (\theta_0, \bar{\theta}_1)$.*

Proof. Consider the dimension of the input (x) is $r \times 1$, the dimension of final weight (θ_1) is $1 \times p$, the dimension of pre-final weight (θ_0) is $p \times r$ and the dimension of pre-final bias (b_0) is $p \times 1$. Let us start by analytically differentiating the neural network:

$$\begin{aligned} y &= \theta_1 \phi(\theta_0 x + b_0) \\ \frac{\partial y}{\partial x} &= \theta_1 \mathsf{diag}(\phi') \theta_0 \\ &\quad (\text{Here, } \phi' = \phi'(\theta_0 x + b_0)) \end{aligned}$$

The dimension of $\frac{\partial y}{\partial x}$ is $1 \times r$, therefore, its transpose will have the dimension of $r \times 1$.

$$\begin{aligned} (\frac{\partial y_l}{\partial x})^T &= (\theta_1 \mathsf{diag}(\phi') \theta_0)^T \\ &= ((\phi')^T \mathsf{diag}(\theta_1) \theta_0)^T \\ (\because \alpha^T \mathsf{diag}(\beta) &= \beta^T \mathsf{diag}(\alpha), \text{if } \alpha, \beta \text{ are same dim vectors}) \\ &= \underbrace{\theta_0^T \mathsf{diag}(\theta_1)}_{\hat{\theta}_l} \phi'(\theta_0 x + b_0) \end{aligned}$$

Now comparing this with the standard neural network we observe that the derivative term ($\frac{\partial y}{\partial x}$) is behaving like a neural network with activation $\hat{\phi} = \phi'_l$ and weight parameters $\hat{\theta} = (\theta_0, \hat{\theta}_1)$ and $\hat{\theta}_1$ is defined as:

$$\hat{\theta}_1 = \theta_0^T \mathsf{diag}(\theta_1)$$

Therefore, the certificate for L-Lipschitz continuity of the derivative term ($\frac{\partial y}{\partial x}$) is given by $M_{\hat{\phi}}(\hat{\theta}, \Lambda) \succeq 0$.
Now, let us analytically calculate $\mathsf{tr}(\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma)$, where σ is $r \times r$ diagonal matrix.

$$\frac{\partial y}{\partial x} = \hat{\theta}_1 \hat{\phi}(\theta_0 x + b_0)$$
$$\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma = \sigma^T (\hat{\theta}_1 \mathsf{diag}(\hat{\phi}')\theta_0)\sigma$$
$$\mathsf{tr}(\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma) = \mathsf{tr}(\sigma^T (\hat{\theta}_1 \mathsf{diag}(\hat{\phi}')\theta_0)\sigma)$$

$$= \underbrace{\begin{bmatrix} \sum_{j=0}^{r} \sigma_j^2 \hat{\theta}_1^{j1} \theta_0^{1j} & \dots & \sum_{j=0}^{r} \sigma_j^2 \hat{\theta}_1^{jp} \theta_0^{pj} \end{bmatrix}}_{\bar{\theta}_1} \phi''(\theta_0 x + b_0)$$

Again, comparing this with the standard neural network we observe that $\mathsf{tr}(\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma)$ is behaving like a neural network with activation $\bar{\phi} = \phi''$ and weight parameters $\bar{\theta} = (\theta_0, \bar{\theta}_1)$ and $\hat{\theta}_1$ is defined as:

$$\bar{\theta}_1 = \begin{bmatrix} \sum_{j=0}^{r} \sigma_j^2 \hat{\theta}_1^{j1} \theta_0^{1j} & \dots & \sum_{j=0}^{r} \sigma_j^2 \hat{\theta}_1^{jp} \theta_0^{pj} \end{bmatrix}$$

Therefore, the certificate for L-Lipschitz continuity of the $\mathsf{tr}(\sigma^T \frac{\partial^2 y}{\partial x^2} \sigma)$ term is given by $M_{\bar{\phi}}(\bar{\theta}, \Lambda) \succeq 0$.

□

Appendix C

Appendix of Chapter 4

C.1 Proofs

C.1.1 Proof of Theorem 4.4.1

Theorem 4.4.1 (Safety quantification of Neural CBF) *Consider a continuous time control system (7.3) for which we obtain a Neural CBF $B_\theta(x)$ parameterized by θ. We sample N i.i.d. samples from $\mathcal{X}$. For a user-specified level α, let $\hat{q}$ be the $\frac{\lceil (N+1)(1-\alpha) \rceil}{N}$th quantile of the conformal scores, $s_j = \max_{i \in \{1,2,3\}}(q_i(x_j))$, on the N state samples. Select a safety violation parameter $\epsilon \in (0,1)$ and a confidence parameter $\beta \in (0,1)$ such that:*

$$\mathcal{I}_{1-\epsilon}(N - l + 1, l) \leq \beta,$$

where $l = \lfloor (N+1)\alpha \rfloor$ and $\mathcal{I}$ is the regularized incomplete beta function. Then, with the probability of at least $1-\beta$, the following holds:

$$\mathbb{P}_{x \in \mathcal{X}} \left(\max_{i \in \{1,2,3\}} (q_i(x)) \leq \hat{q} \right) \geq 1 - \epsilon.$$

Proof. To lay the foundation for our methodology, we first introduce Split Conformal Prediction:

Lemma C.1 (Split Conformal Prediction [10]). *Consider a set of independent and identically distributed (i.i.d.) calibration data, denoted as* $\{(X_i, Y_i)\}_{i=1}^n$, *along with a new test point* (X_{test}, Y_{test}) *sampled independently from the same distribution. Define a score function* $s(x, y) \in \mathbb{R}$, *where higher scores indicate poorer alignment between* x *and* y. *Compute the calibration scores* $s_1 = s(X_1, Y_1), \ldots, s_n = s(X_n, Y_n)$. *For a user-defined confidence level* $1 - \alpha$, *let* $\hat{q}$ *represent the* $\lceil (n+1)(1-\alpha) \rceil / n$ *quantile of these scores. Construct the prediction set for the test input* X_{test} *as:*

$$\mathcal{C}(X_{test}) = \{y : s(X_{test}, y) \leq \hat{q}\}.$$

Assuming exchangeability, the prediction set $\mathcal{C}(X_{test})$ *guarantees the marginal coverage property:*

$$\mathbb{P}(Y_{test} \in \mathcal{C}(X_{test})) \geq 1 - \alpha.$$

Remark C.1.1. *The coverage of conformal prediction, conditioned on the calibration set, is a random variable. Running the algorithm with different calibration sets results in varying coverage over an infinite validation set. While Lemma C.1 ensures coverage is at least* $1 - \alpha$ *on average, it deviates for any fixed calibration set. However, increasing* n *reduces these fluctuations, and the coverage distribution follows an analytic form, as derived by Vladimir Vovk in [174]:*

$$\mathbb{P}(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})) \sim \text{Beta}(n + 1 - l, l), \tag{C.1}$$

where $l = \lfloor (n+1)\alpha \rfloor$.

Using Lemma C.1, we define the input as $X = x$ and the corresponding output as $Y = \max_{i \in \{1,2,3\}}(q_i(x))$. The conformal scoring function is then given by $s(X, Y) := \max_{i \in \{1,2,3\}}(q_i(x))$, where higher scores correspond to greater misalignment between x and y, or equivalently, indicate higher safety violations in our setting.
Accordingly, the prediction set for a test input $X_{\text{test}} = x$ is defined as:

$$\mathcal{C}(X_{test} = x) = \{y : \max(q_i(x)) \leq \hat{q}\}, \ \forall x \in \mathcal{X}.$$

Now, using eq (C.1), we get

$$\mathop{\mathbb{P}}_{x \in \mathcal{X}} \left(\max_{i \in \{1,2,3\}} (q_i(x)) \leq \hat{q} \right) \sim \text{Beta}(N + 1 - l, l).$$

Define E as:

$$E := \mathop{\mathbb{P}}_{x \in \mathcal{X}} \left(\max_{i \in \{1,2,3\}} (q_i(x)) \leq \hat{q} \right).$$

Here, E is a Beta-distributed random variable. Using properties of CDF, we assert that $E \geq 1 - \epsilon$ with confidence $1 - \beta$ if the following condition is satisfied:

$$\mathcal{I}_{1-\epsilon}(N - l + 1, l) \leq \beta, \tag{C.2}$$

where $\mathcal{I}_x(a, b)$ is the regularized incomplete Beta function and also serves as the CDF of the Beta distribution.
Thus, if Equation (C.2) holds, we can conclude with probability $1 - \beta$ that:

$$\mathop{\mathbb{P}}_{x \in \mathcal{X}} \left(\max_{i \in \{1,2,3\}} (q_i(x)) \leq \hat{q} \right) \geq 1 - \epsilon.$$

□

C.2 Relationship between α, β, and ϵ

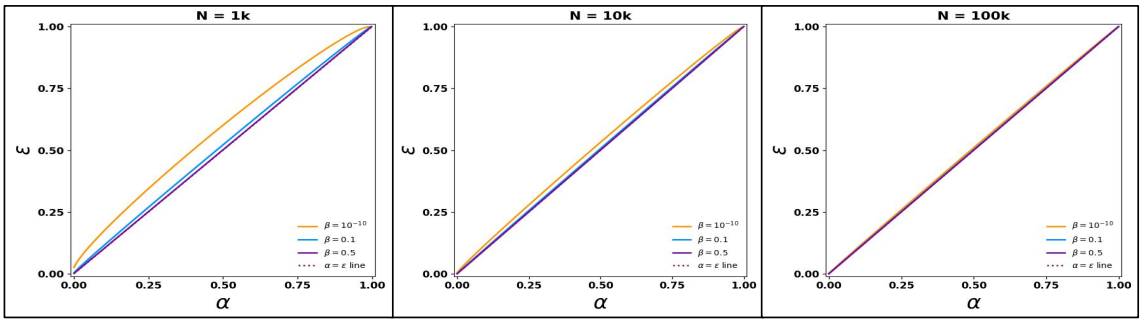


Figure C.1: This figure shows the α-ϵ plots for different numbers of verification samples, N, and different values of β.

The work [174] states that a smaller number of samples leads to greater fluctuations in the conformal prediction calibration, meaning that if we redraw N samples and repeat the conformal prediction process, we might get a different calibration result. This variance decreases as N increases.Similarly, in our work, a small N means that the value correction term δ might fluctuate each time the verification algorithm is executed. Therefore, to ensure a stable estimate of δ, it is desirable to select a sufficiently large value of N.

Figure C.1 presents the $\alpha - \epsilon$ plots for varying numbers of verification samples N and different values of β. From the figure, we observe that as N increases, the effect of β diminishes, and the curve approaches the $\alpha = \epsilon$ line. Ideally, the user-specified safety error rate (α) should closely match the safety violation parameter (ϵ) while

maintaining high confidence ($1-\beta$ close to 1). Thus, selecting a larger N enables a smaller β while ensuring the alignment of α and ϵ. Conversely, if N is small, one must either compromise on the confidence parameter β or accept that α will be lower than ϵ, resulting in a more conservative upper bound on the safety rate.

C.3 Experimental Details

C.3.1 Autonomous Ground Vehicle Collision Avoidance

We consider an autonomous ground vehicle collision avoidance problem with the state space given by $\mathcal{X} = [-2, 2]^2 \times [-\pi, \pi]$. The safe region given by $\mathcal{X}_s = \mathcal{X} \backslash [-1.5, 1.5]^2 \times [-\pi, \pi]$ and the unsafe region by $\mathcal{X}_u = [-0.2, 0.2]^2 \times [-\pi, \pi]$.

C.3.2 Aerial Vehicle Geo-Fencing

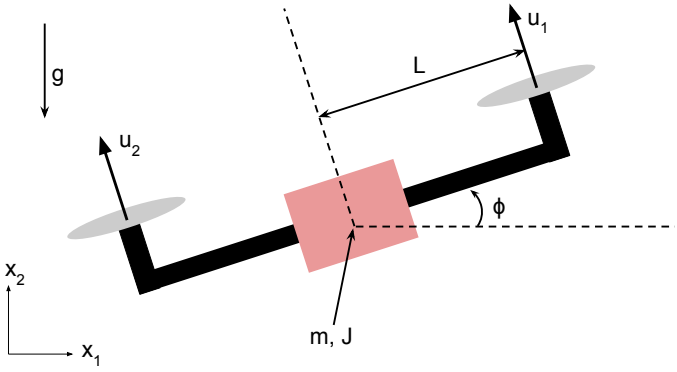


Figure C.2: A visual representation of the aerial vehicle

The control-affine dynamics of a planar aerial vehicle are given as:

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & -\frac{1}{m}\sin\phi \\ \frac{1}{m}\cos\phi & \frac{1}{m}\cos\phi \\ \frac{1}{LJ} & -\frac{1}{LJ} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{C.3}$$

where $[x_1, x_2, \phi, \dot{x}_1, \dot{x}_2, \dot{\phi}]^T \in \mathcal{X} \subseteq \mathbb{R}^6$ represents the state vector comprising the position (x_1, x_2) and orientation ϕ of the vehicle. The inputs u_1 and u_2 correspond to the thrusts generated by the right and left motors, respectively, with mass m, moment of inertia J, and motor separation length $2L$. The gravitational constant is g. We normalize the parameters to $m = 1$ and $1/LJ = 1$ for simplicity.

We use a PD controller to generate a desired controller (u_{des}), which tracks the desired trajectory. The PD controller formulated for the quadrotor is given as follows:

$$\begin{bmatrix} \ddot{x_1} \\ \ddot{x_2} \end{bmatrix} := \left(\begin{bmatrix} \ddot{x_{1d}} \\ \ddot{x_{2d}} \end{bmatrix} + \begin{bmatrix} K_{1d}(\dot{x_{1d}} - \dot{x_1}) \\ K_{2d}(\dot{x_{2d}} - \dot{x_2}) \end{bmatrix} + \begin{bmatrix} K_{1p}(x_{1d} - x_1) \\ K_{2p}(x_{2d} - x_2) \end{bmatrix} \right)$$

$$\begin{bmatrix} \ddot{\phi} \end{bmatrix} := \left(\begin{bmatrix} K_{rd}(\dot{\phi_d} - \dot{\phi}) \end{bmatrix} + \begin{bmatrix} K_{rp}(\phi_d - \phi) \end{bmatrix} \right)$$

where, $\phi_d = -\frac{\ddot{x_1}}{g + \ddot{x_2}}$

$$\begin{bmatrix} (u_1)_{ref} \\ (u_2)_{ref} \end{bmatrix} := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} m\sqrt{(\ddot{x_1}^2 + (g + \ddot{x_2})^2)} \\ LJ\ddot{\phi} \end{bmatrix}$$

where, $K_p's$ and $K_d's$ are the proportional and derivative coefficients of respective variables.

The state space is defined as $\mathcal{X} = [-2, 2]^2 \times [-\pi, \pi] \times [-2, 2]^3$. The safe region is $\mathcal{X}_s = [-0.8, 0.8]^2 \times [-\pi, \pi] \times [-2, 2]^3$, while the unsafe region is $\mathcal{X}_u = \mathcal{X} \backslash [-1, 1]^2 \times [-\pi, \pi] \times [-2, 2]^3$.

Simulation Results

We evaluated the CP-NCBF-QP controller (Eq. (7.7)) at a 99.5% safety level in a UAV environment built on the Pybullet physics engine [39] using the Pybullet-Drones framework [114]. The vehicle is required to track a predefined reference trajectory while ensuring it remains within the geofenced interval $x_1 \in [-1, 1]$.

Figure C.3 illustrates these results: the red curve corresponds to the reference trajectory, the grey vertical lines at $x_1 = -1$ and $x_1 = 1$ mark the boundaries, and the blue curve shows the trajectory actually executed by the safe controller on a Crazyflie drone in simulation.

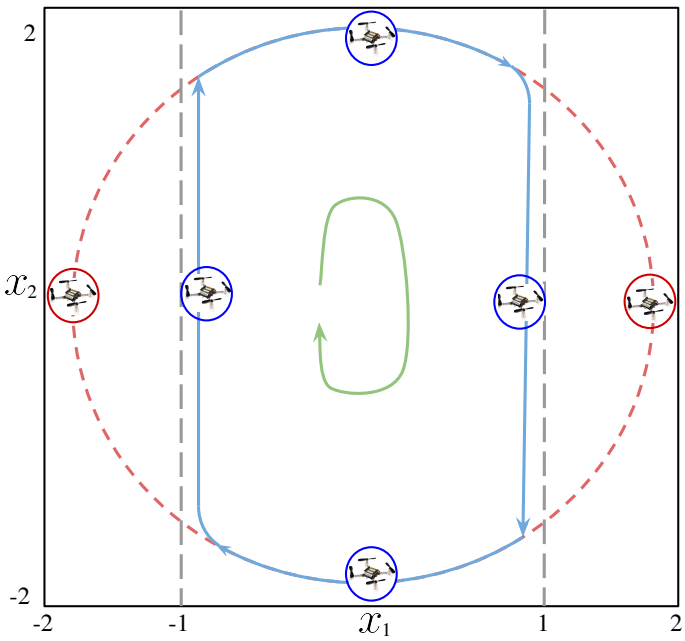


Figure C.3: Implementation of the CP-NCBF-QP controller on a Crazyflie drone in Pybullet. The red line denotes the desired reference path, grey lines indicate the safety limits at $x_1 = -1$ and $x_1 = 1$, and the blue line represents the safe trajectory generated by our controller.

C.3.3 Quadruped Navigation in Dynamic Environment

The control-affine dynamics of a reduced order Quadruped moving in a dynamic environment is given as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\phi} \\ \dot{x}_{o1} \\ \dot{x}_{o2} \\ \dot{v}_{o1} \\ \dot{v}_{o2} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_{o1} \\ v_{o2} \\ k_1 \\ k_2 \\ k_r \end{bmatrix} + \begin{bmatrix} cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{C.4}$$

where $[x_1, x_2, \phi, x_{o1}, x_{o2}, v_{o1}, v_{o2}, r]^T \in \mathcal{X} \subseteq \mathbb{R}^8$ represents the state vector comprising the position (x_1, x_2) and orientation ϕ of the quadruped, (x_{o1}, x_{o2}) represents the position of the obstacle, (v_{o1}, v_{o2}) represents the velocity of the obstacle and r represents the radius of the obstacle. The inputs u_1 and u_2 correspond to the speed and the rate of change of orientation of the quadruped, respectively.

The state space is defined as $\mathcal{X} = [-2, 2]^2 \times [-\pi, \pi] \times [-2, 2]^2 \times [-1, 1]^2 \times [0.5, 1]$. In this case we sample the safe and unsafe states using the concept of collision cone

[27, 26].

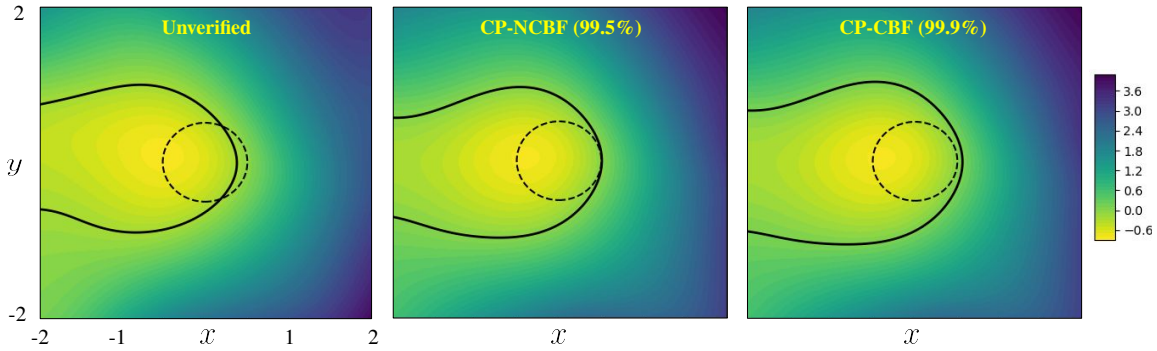


Figure C.4: Visualization of learned NCBFs for the quadruped navigation at $\phi, x_{o1}, x_{o2}, v_{o1}, v_{o2} = 0$, $r = 0.5$: (Left) Unverified NCBF (without robust loss), (Center) CP-NCBF (with 99.5% safety guarantee), and (Right) CP-NCBF (with 99.9% safety guarantee). Bold black lines denote the zero-level sets.

Comparison of CP-NCBF with Baselines

The Figure C.4 shows the visualization of the learned NCBFs for quadruped navigation in dynamic environments.

Simulation Results

We evaluated the CP-NCBF-QP controller (Eq. (7.7)) at a 99.5% safety level in a quadruped locomotion task implemented on the Pybullet physics engine [39] via the Motion Imitation repository[1]. In this setup, the quadruped is commanded to follow a straight-line trajectory while dynamically avoiding a moving obstacle.

Figure C.5 presents a sequence of snapshots capturing the quadruped as it safely negotiates the moving obstacle under the CP-NCBF-QP controller's guidance.

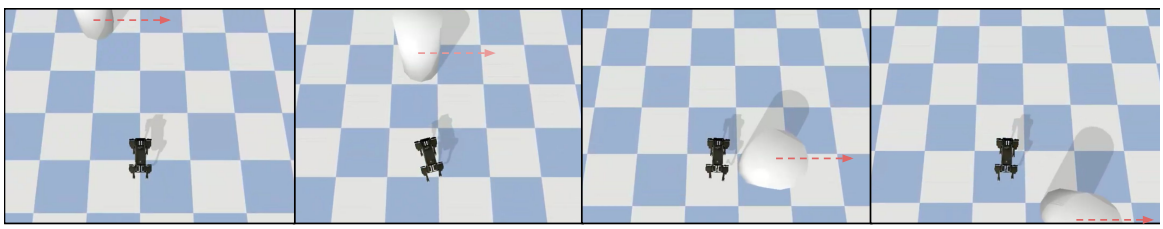


Figure C.5: Snapshots of quadruped navigation in Pybullet with a moving obstacle. The CP-NCBF-QP controller successfully prevents collisions while the agent progresses along its straight-line path.

[1] https://github.com/erwincoumans/motion_imitation

C.4 Implementation Details of the Algorithms

This section provides an in-depth overview of our algorithm and baseline implementations, including hyperparameter configurations and the cost/reward functions used in the baselines across all experiments.

C.4.1 Experimental Hardware

All experiments were conducted on a system equipped with an 11th Gen Intel Core i9-11900K @ 3.50GHz × 16 CPU, 128GB RAM, and an NVIDIA GeForce RTX 4090 GPU for training.

C.4.2 Hyperparameters for the Proposed Algorithm

The relative importance weights λ_1, λ_2 (in eq.4.4) are 1 and 0.1, respectively. The training settings for all the experiments are detailed in the Table C.1.

Hyperparameter	**Value**
Network Architecture	Multi-Layer Perceptron (MLP)
Activation Function	Softplus ($\log(1+\exp(x))$)
Optimizer	Adam optimizer
Learning Rate	1×10^{-3}
Autonomous Ground Vehicle Collision Avoidance	
Number of Hidden Layers	1
Hidden Layer Size	64 neurons per layer
Number of Training Points	20K
No. of Training Epochs	1000
Aerial Vehicle Geo-Fencing	
Number of Hidden Layers	2
Hidden Layer Size	128 neurons per layer
Number of Training Points	20K
No. of Training Epochs	1500
Quadruped Navigation in dynamic environment	
Number of Hidden Layers	2
Hidden Layer Size	128 neurons per layer
Number of Training Points	50K
No. of Training Epochs	5000

Table C.1: Hyperparameters for the proposed algorithm

C.4.3 Hyperparameters for the Baseline

For the Lipschitz-constrained NCBF method of [165] (designed for autonomous ground vehicle collision avoidance), we adopted their publicly available implementation[2] and set $\sigma = 0$ to enforce the deterministic regime. Training required $N = 54$M data points sampled densely across the state–action space; this sampling strategy, however, proved intractable for higher-dimensional tasks such as aerial vehicle geofencing and dynamic quadruped navigation.

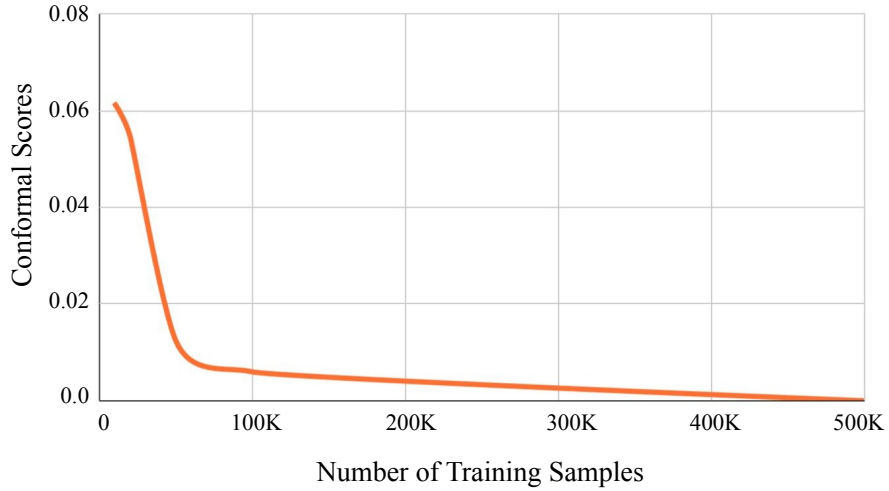
We also implemented the unverified neural CBF formulation of [46], using their open-source codebase[3]. In this case, we matched all hyperparameters to those specified for our approach (see Table 1 for details).

C.4.4 Conformal Scores vs. Training Set Size

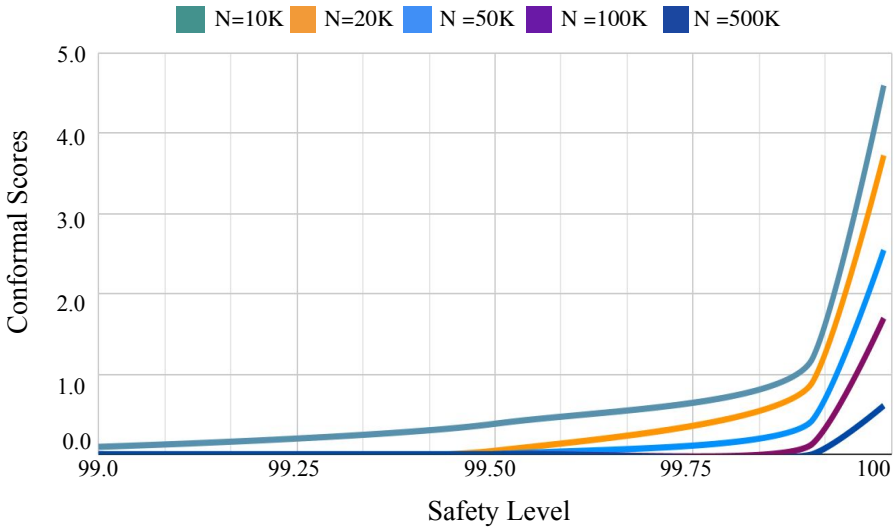
We conducted an ablation study to investigate how the number of training samples affects the conformal safety guarantees of NCBFs in two settings: autonomous ground vehicle collision avoidance and aerial vehicle geo-fencing.

Figure C.6 summarizes our findings. In subfigure C.6a, we plot the conformal scores of ground-vehicle NCBFs trained on varying dataset sizes, all evaluated at the 99.9% safety level. Subfigure C.6b shows the conformal scores for aerial-vehicle NCBFs as a function of training sample size and different target safety levels. It can be inferred that the conformal scores decrease as the sample size increases and rise when stricter safety guarantees are imposed.

[2] https://github.com/tayalmanan28/Stochastic-NCBF
[3] https://github.com/MIT-REALM/neural_clbf



(a) Ground vehicle: conformal score vs. number of training samples at 99.9% safety.



(b) Aerial vehicle: conformal score vs. training samples for various safety levels.

Figure C.6: Dependence of conformal safety scores on the size of the training dataset for (a) ground-vehicle collision avoidance and (b) aerial-vehicle geo-fencing tasks.

Appendix D

Appendix of Chapter 7

D.1 Proofs

D.1.1 Finite Difference Barrier Condition

Lemma D.1 (Finite-Difference Approximation of the Barrier Condition)**.** *Consider a control-affine system $\dot{x} = f(x) + g(x)u$ with a continuously differentiable barrier function $B : \mathcal{X} \to \mathbb{R}$. The continuous-time barrier condition*

$$\min\{\max_{u} \nabla B(x)^\top (f(x) + g(x)u),\ \ell(x) - B(x)\} = 0 \tag{D.1}$$

admits the finite-difference approximation

$$\min\{\max_{u_t} B(x_{t+1}),\ \ell(x_t)\} = B(x_t), \tag{D.2}$$

up to a positive scaling by Δt.

Proof. Starting from the continuous-time condition in (D.1), note that

$$\nabla B(x)^\top (f(x) + g(x)u) = \nabla B(x)^\top \dot{x} = \frac{d}{dt} B(x(t)). \tag{D.3}$$

Using a forward Euler approximation, the time derivative satisfies

$$\frac{d}{dt} B(x(t)) \approx \frac{B(x_{t+1}) - B(x_t)}{\Delta t}, \tag{D.4}$$

where, $x_t = x(t)$, $u_t = u(t)$ and $x_{t+1} = x(t + \Delta t)$. Substituting (D.4) into (D.1)

yields

$$\min\left\{\max_{u_t}\frac{B(x_{t+1})-B(x_t)}{\Delta t},\ \ell(x_t)-B(x_t)\right\}=0. \tag{D.5}$$

Let $a = B(x_{t+1}) - B(x_t)$ and $b = \ell(x_t) - B(x_t)$. Since $\Delta t > 0$, the function

$$m(a,b) = \min\{\, a/\Delta t,\ b\,\}$$

satisfies $m(a,b) = 0$ if and only if $\min\{a,b\} = 0$. This follows from the fact that scaling one argument of the minimum by a positive constant does not change which argument is smaller, nor the value at which the minimum equals zero. Thus, (D.5) is equivalent to

$$\min\{\, a,\ b\,\} = 0, \tag{D.6}$$

which gives the finite-difference barrier condition in (D.2). This establishes the claim. □

D.1.2 Expectile Regression

Expectile regression is a classical tool in statistics and econometrics for estimating asymmetric statistics of a random variable. For a random variable X, the τ-expectile is defined as the minimizer of the asymmetric least-squares problem

$$m_\tau = \arg\min_m \mathbb{E}_{x\sim X}\left[L^\tau(x-m)\right], \tag{D.7}$$

where $L_\tau(y) = |\tau - \mathbf{1}(y<0)|, y^2$.

When $\tau > 0.5$, this loss assigns more weight to samples above the estimate m_τ and less weight to samples below it. Conversely, $\tau < 0.5$ emphasizes lower values. Thus, expectiles interpolate smoothly between the mean ($\tau = 0.5$) and a "high-value–seeking" statistic as $\tau \to 1$.

Expectile regression can also be extended to learning conditional expectiles:

$$m_\tau(x) = \arg\min_{m(\cdot)} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[L^\tau(y-m(x))\right], \tag{D.8}$$

which can be optimized efficiently via stochastic gradient descent. This makes expectiles easy to implement in modern machine-learning pipelines, unlike alternative high-order statistics that require specialized solvers.

Why Expectile Regression in V-OCBF?

In V-OCBF, the barrier function ideally requires computing $\max_{u' \in \mathcal{U}} B(x')$, but performing this maximization directly is problematic in the offline setting because the dataset supports only a restricted action set $\mathcal{U}_{\mathcal{D}} = \{ u \mid (x, u, x') \in \mathcal{D} \}$, and querying $B(x')$ for unseen actions $u' \notin \mathcal{U}_{\mathcal{D}}$ introduces unsupported targets that can distort or destabilize learning. Moreover, even restricting the maximization to $\mathcal{U}_{\mathcal{D}}$ leads to a brittle update since a hard maximum is extremely sensitive to noise or rare outlier transitions, often producing poor barrier estimates. Expectile regression provides a smooth and data-supported alternative by emphasizing higher (safer) barrier values when $\tau > 0.5$, effectively approximating the maximal barrier value over $\mathcal{U}_{\mathcal{D}}$ while never querying out-of-distribution actions, thereby yielding a stable and principled mechanism for synthesising offline CBVF.

Expectile regression resolves these issues. For $\tau > 0.5$, the expectile estimator concentrates on the largest barrier values consistent with the dataset. This yields a smooth approximation to $\max_{u' \in \mathcal{U}_{\mathcal{D}}} B(x')$, which is equivalent to performing a constrained maximization over only the dataset-supported actions, without ever querying B on out-of-distribution actions.

To illustrate the effect of the expectile parameter we show barrier functions learned for the AGV case study using $\tau \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$ (Figure D.1). As τ increases the volume of the sub-zero level set visibly shrinks: lower τ values produce larger sub-zero regions because the learning target is more influenced by poorer next-state outcomes present in the data, which inflates the set of states that appear unsafe under the learned barrier. By contrast, larger τ up-weights higher (safer) barrier targets within the dataset and yields a smoother, upper-envelope–like approximation to the maximization over dataset-supported actions; sampling actions from this higher expectile therefore produces a smaller sub-zero level set (i.e., a tighter unsafe region) and correspondingly larger estimated safe set.

D.1.3 Feasibility of V-OCBF-QP

Formal safety guarantees of standard QP-based CBF formulations generally hold under the assumption of unbounded actuation. In practice, however, in the presence of control bounds, a nominal CBF may conflict with feasibility requirements, causing the corresponding CBF-QP to become infeasible. To address this within our framework, we adopt a standard relaxation technique from the safe control literature by introducing a slack variable δ. This relaxes the hard safety constraint when strictly necessary

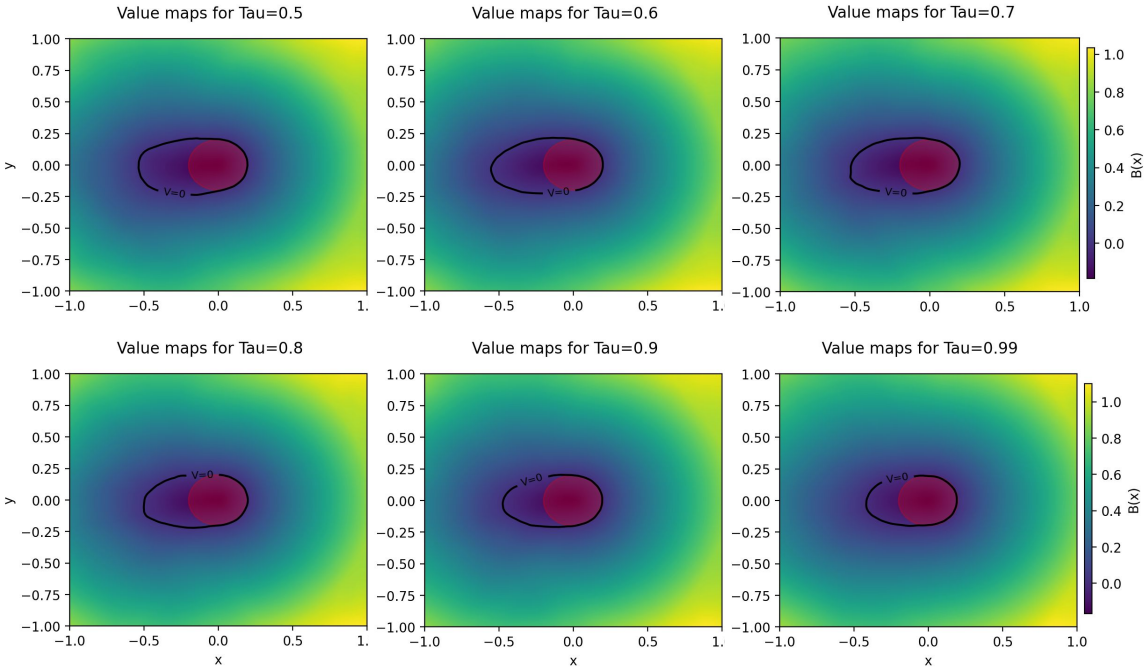


Figure D.1: **τ-Expectiles:** Figure shows a depiction of the learned Barrier Function for different τ values with Expectile Regression. With increasing value of τ (> 0.5), the feasible region increases, indicating a more accurate representation of the true barrier function.

while heavily penalizing any violations. The resulting slack-augmented V-OCBF-QP is formulated as:

$$\begin{aligned} \pi_{\text{safe}}(x) = \arg\min_{u \in \mathbb{U}, \delta \in \mathbb{R}} \|u - \pi_{\text{ref}}(x)\|^2 + \lambda \delta^2 \\ \text{s.t. } \mathcal{L}_f B(x) + \mathcal{L}_g B(x) u + \kappa\left(B(x)\right) + \delta \geq 0, \end{aligned} \tag{D.9}$$

where $\lambda > 0$ is a penalty coefficient.

Empirically, δ remains close to zero throughout our experiments. Because V-OCBF explicitly accounts for control constraints during learning, the synthesized policy naturally respects admissible bounds. Consequently, the safety filter rarely becomes infeasible, and any triggered corrective adjustments require minimal slack usage.

D.2 Description of the Experiments

D.2.1 Autonomous Ground Vehicle Collision Avoidance

We consider a collision-avoidance task for an autonomous ground vehicle modeled using the Dubins dynamics:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} v\cos\Phi \\ v\sin\Phi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{D.10}$$

where the state $x = [x_1, x_2, \phi]^T \in \mathcal{X} \subseteq \mathbb{R}^3$ encodes the vehicle's position and heading. The forward velocity is fixed at $v = 0.6$ units/s, and the admissible state space is given by $\mathcal{X} = [-1, 1]^2 \times [-\pi, \pi]$. The control input $u \in [-1, 1]$ denotes the angular velocity ω.

The reward function $r(x)$ is defined as

$$r(x) = \frac{C}{\left(\left\| \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} x_{g1} \\ x_{g2} \end{bmatrix} \right\| + \epsilon \right)} \tag{D.11}$$

where $C = 0.1$ is the scaling factor and $(x_{g1}, x_{g2})^T$ denotes the goal location. Rewards are accumulated only while the agent remains collision-free. The constant $\epsilon > 0$ is included to prevent numerical instability when the vehicle approaches the goal.

Safety is encoded through the function

$$\ell(x) = \left\| \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} x_{o1} \\ x_{o2} \end{bmatrix} \right\| - R, \tag{D.12}$$

where $(x_{o1}, x_{o2})^T = (0, 0)^T$ denotes the center of the obstacle and $R = 0.2$ units is the radius of the obstacle.

Offline Data Generation. Since this is a custom environment, we construct an offline dataset for training both the barrier function and the safe policy. We sample 1500 initial states uniformly from $\mathcal{X}$ and simulate each trajectory for 500 discrete timesteps with step size $\Delta t = 0.01s$. During data collection, control inputs are drawn uniformly at random from the admissible range, ensuring diverse system trajectories for learning-based safety certification.

Safe Set Volume Analysis

Backward Reachable Tube is the set of initial states for which the agent acting optimally, will eventually reach the target set $\mathcal{T}$ within the time horizon $[0, t]$:

$$SSV(x) = \{x : \exists u(\cdot), \exists t \in [0, T], \xi^{u}_{x,0}(t) \in \mathcal{T}\} \tag{D.13}$$

where $\xi^{u}_{x,0}(t)$ represents system state at time $t \in [0, T]$, and u defined over time horizon $[0, T]$, represents sequence of control actions over the time horizon.

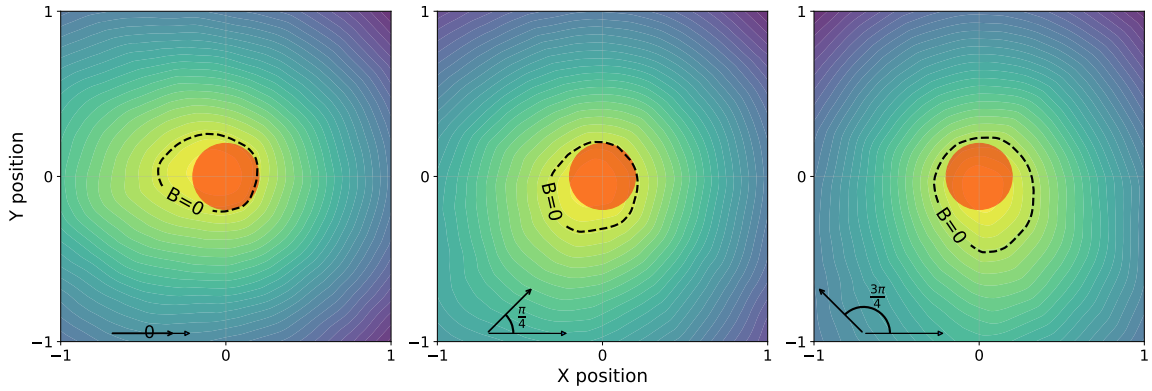


Figure D.2: **Safe Set Volume:** Figure shows a depiction of feasible region (i.e., anything outside B=0 enclosure) for V-OCBF for 3 different heading angles. Note that calculation of the actual Safe Set volume is a cumulative of feasible region over countless such frames along the continuous heading angle (ϕ) dimension.

We performed Safe Set Volume (SSV) analysis to inspect the effectiveness and conservatism of various approaches. Specifically, we aim to evaluate the volume of the state space from which the agent can safely initialize. Effectively, the larger this Safe Set volume is, bigger the feasible region gets, hence, making it easy to navigate in the environment, and smaller the Safe Set volume becomes, lesser the feasible region gets, thus, overall making it less safe for the agent to navigate in the environment.

To evaluate the safe initialization region that each of the method casts, we uniformly sample different initial states across the state space and check whether they belong to the above-defined Safe Set Volume by rolling out trajectories in the online environment for over 100k times (as we started to observe convergence in the Safe Set Volume after the 100k mark). We investigate the results for this analysis in the main paper (refer Table 7.1) where we show that V-OCBF enables the largest safe feasible region among all the various approaches.

D.2.2 Safety MuJoCo Environments:

To evaluate our framework on higher-dimensional systems, we use MuJoCo Safety Gymnasium environments [77], where safe velocity tasks introduce velocity constraints for Gymnasium's MuJoCo-v4 agents.

The agent receives a scalar cost signal on the basis of its velocity at each step:

$$cost = bool(V_{current} > V_{threshold}) \tag{D.14}$$

This constraint can be framed as a safety function

$$\ell(x) = V_{threshold} - V_{current} \tag{D.15}$$

since $\ell(x) \geq 0$ corresponds to safe state. This safety value function provides us a continuous signal instead of sparse values of 0 or 1. For agent specific threshold velocities and time-step Δt, we refer the official documentation, the official values for which have been compiled and included in Table D.1.

Table D.1: Threshold velocity $V_{threshold}$, time step size Δt and action space $\mathbb{U}$ for each MuJoCo based Safety Gymnasium [77] environment as per official documentation.

Environment	**Hopper**	**Half-Cheetah**	**Ant**	**Walker2D**	**Swimmer**
$\boldsymbol{V_{threshold}}$	0.7402	3.2096	2.6222	2.3415	0.2282
$\boldsymbol{\Delta t}$	0.008	0.05	0.05	0.008	0.04
$\mathbb{U}$	$[-1.0, 1.0]^3$	$[-1.0, 1.0]^6$	$[-1.0, 1.0]^8$	$[-1.0, 1.0]^6$	$[-1.0, 1.0]^2$

Qualitative Analysis

In continuation to the Hopper plot in the main paper (Fig. 7.4), we include plots for all the remaining MuJoCo safety gymnasium environments covered in this paper with the visualization of the episode rollouts, both with and without V-OCBF. Given the safety rates of V-OCBF compared to the other baselines (refer Fig. 7.3), it becomes important to visually realize the real-time effect on performance (Figures D.3, D.4, D.5).

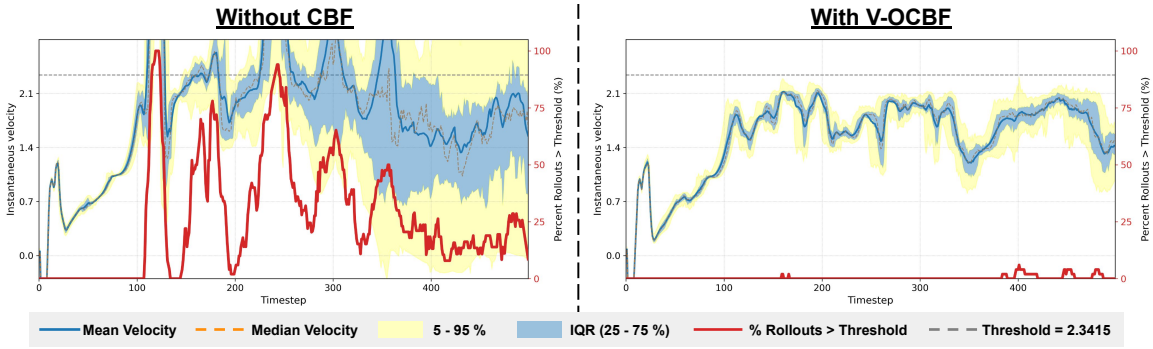


Figure D.3: **Walker-2D.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep.

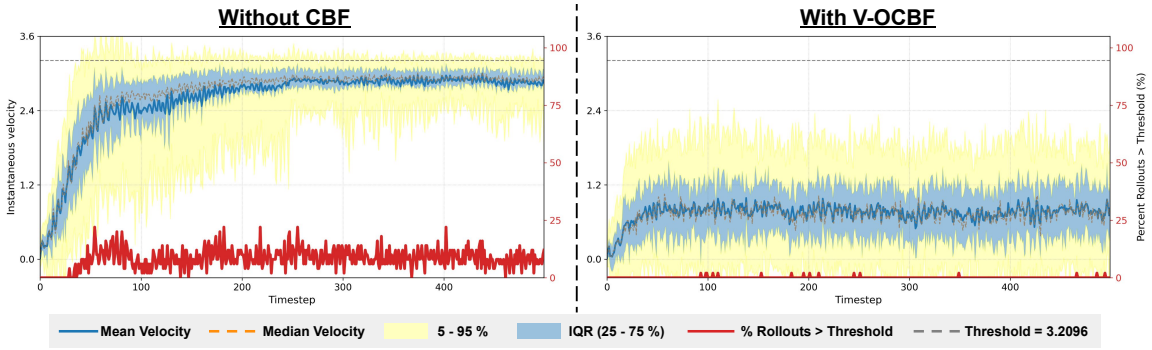


Figure D.4: **Half-Cheetah.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep.

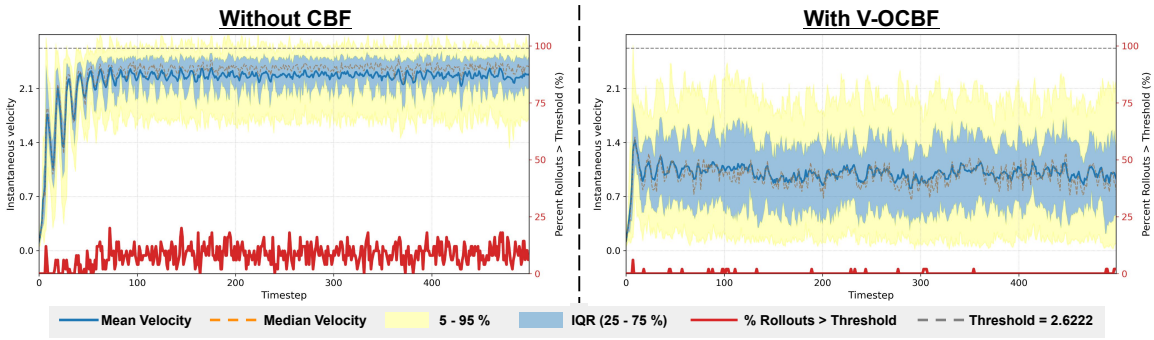


Figure D.5: **Ant.** Instantaneous velocity (left Y-axis) across episodes. Right Y-axis depicts percentage of episodes when the agent violates the velocity threshold at a particular timestep.

D.3 Ablation Studies

We have performed a set of ablation studies on our primary testing environment of Autonomous Ground Vehicle Collision Avoidance to further strengthen our claims of

the paper. AGV acts as an ideal testing bed for such ablations as it is easy to visualize and infer from the results in a 2D space. Besides, the environment properly encapsulates our requirements of providing a safety critical control environment. Following subsections cover the ablations we have performed.

D.3.1 Robustness to size of the NN

In this section we show our framework's robustness against varying model sizes, thus, demonstrating the versatility on different hyperparameter choices. Table D.2 shows the safety rate for different model dimensions.

Table D.2: Safety Rate (%) reported for various model dimensions. Columns represent Width: neurons in each hidden layer, and Rows represent number of layers.

Width → **Layers ↓**	**256** **Neurons**	**96** **Neurons**	**64** **Neurons**	**48** **Neurons**
2	**98.28 ± 0.54**	97.18 ± 0.41	96.96 ± 1.21	–
3	98.04 ± 0.30	97.36 ± 0.22	97.12 ± 0.23	96.54 ± 1.62
4	–	–	–	96.82 ± 1.24

As can be seen from the Table D.2, the models don't affect the safety performance significantly, until we drop the size by too much (as in the cases of **(48,3)**, **(48,4)** and **(64,2)**). We consider the subtle drop is safety rate for the lower dimensional networks to be within considerable limits.

D.3.2 Robustness to errors in Learned Dynamics

To evaluate the robustness of our pipeline to errors in the learned dynamics, we measure how additive perturbations affect the safety performance of the CBF-based controllers. Concretely, we perturb the computed safe action ω by adding zero-mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ with a range of standard deviations σ (refer D.3). For every noise level we run 500 independent trials (with 5 different random seeds each) and report the safety rate defined as the fraction of trials that remain within the safe set over the evaluation horizon. Table D.3 summarizes the results across several CBF variants, for each method we report mean safety rate ± standard error. Across noise levels spanning small to large perturbations, the safety rates degrade only mildly, indicating that the learned CBFs tolerate moderate inaccuracies in the dynamics / control signal. These results suggest that, while precise dynamics models improve performance, our approach provides substantially safety-aware control even when the dynamics model is imperfect.

Table D.3: Safety Rate (%) reported for Robustness Testing of **learned Dynamics** for Autonomous Ground Vehicle by adding upto 5%, 10% and 20% of the possible control action as noise.

Method	**No Noise**	**+ Small Noise (5%)**	**+ Medium Noise (10%)**	**+ Large Noise (20%)**
BC+NCBF [132]	92.48 ± 0.60	92.48 ± 0.62	92.52 ± 0.41	92.32 ± 0.56
BC+iDBF [25]	92.87 ± 0.73	92.84 ± 0.64	92.40 ± 0.84	92.76 ± 0.28
BC+CCBF [151]	93.56 ± 0.56	93.72 ± 0.58	93.68 ± 0.80	93.76 ± 0.38
BC+**V-OCBF** (Ours)	**98.28 ± 0.54**	**98.08 ± 0.61**	**97.84 ± 0.68**	**97.94 ± 0.48**

This empirical test shows that even if we haven't learned the most accurate dynamics model, we can expect a considerably precise safety-aware control for the system with our approach.

Sensitivity to Expectile parameter (τ)

As discussed in Section 7.4.2, we use a τ-expectile loss to obtain the final CBVF. When $\tau = 0.5$, the objective reduces to a symmetric MSE fit, causing the learned barrier to reflect the average safety value induced by the behavior policy. This reproduces the *behavior-induced* barrier and typically yields overly conservative safe sets that do not generalize well beyond the demonstrated trajectories.

To investigate this effect, we conduct a study by comparing $\tau = 0.5$ against higher expectile levels that emphasize the upper tail of the safety value distribution. Table D.4 summarises the safety rates achieved on the AGV environment for different τ values:

Table D.4: Safety Rate (%) reported for Autonomous Ground Vehicle by varying τ in the expectile regression.

τ	**0.5 (Behavior Policy)**	**0.7**	**0.8**	**0.9**	**0.99**
Safety Rate (%)	96.53 ± 0.23	96.87 ± 0.54	97.16 ± 1.01	97.72 ± 0.88	**98.28 ± 0.54**

The results reveal two key patterns. First, the behavior-induced barrier ($\tau = 0.5$) yields the lowest safety rate, consistent with its tendency to underrepresent safety-critical regions that are not frequently explored by the behavior policy. Second, as τ increases, the barrier becomes progressively more conservative in safety-critical areas and better aligned with the upper end of the demonstrated safety values, resulting in consistently higher safety rates. Figure D.1 depicts this nature of the learned barrier functions on varying the value of τ.

Under idealized conditions (deterministic dynamics and no dataset disturbances ,i.e., when the best actions reliably lead to the best next states), pushing the expectile

$\tau \to 1$ concentrates the estimator on the upper tail of the target distribution and, under favorable (nearly deterministic) conditions, can yield the strong empirical performance, prioritizing high-value next-state targets. However, this extreme setting is brittle to rare poor actions or stochastic disturbances that produce optimistic next states; in such cases an estimator that strictly targets the extreme upper tail may overfit those spurious outcomes.

To empirically validate this reasoning, we conducted an additional set of experiments evaluating the impact of τ across multiple Safety MuJoCo tasks using standard, public datasets from DSRL, which are not strictly deterministic. We evaluated both safety (measured via Cost) and task performance (measured via Reward) for $\tau \in \{0.5, 0.7, 0.9, 0.99\}$, as shown in Table D.5.

Table D.5: Reward and Cost across different τ expectile values for selected MuJoCo environments

	$\tau = 0.5$		$\tau = 0.7$		$\tau = 0.9$		$\tau = 0.99$	
Environment	Reward	Cost	Reward	Cost	Reward	Cost	Reward	Cost
Ant	565.36	0.0	639.61	0.0	791.77	0.0	734.66	0.02
Half Cheetah	100.39	0.0	180.87	0.0	679.73	0.0	582.58	0.05
Hopper	96.94	0.0	126.54	0.0	769.69	0.0	526.72	0.27

As observed in Table D.5, increasing τ toward 1 (specifically $\tau = 0.99$) results in degraded safety performance, reflected by increased Cost, and in some cases reduced Reward. This trend is consistent across the reported tasks and aligns with our theoretical intuition regarding sensitivity to stochastic transitions. In contrast, $\tau = 0.9$ achieves a more favorable balance between reward maximization and safety preservation. Similar patterns were observed across other environments. Based on these findings, we selected $\tau = 0.9$ for the experiments reported in the main paper.

D.3.3 Sensitivity to Dynamics Model Accuracy

To empirically evaluate the resilience of our framework against approximation errors in the learned dynamics, we conducted an analysis using the Autonomous Ground Vehicle environment to quantify how variations in model fidelity influence safety performance. We evaluated four distinct configurations representing varying levels of accuracy: the known system dynamics, the standard learned model used in our main experiments, and two degraded models trained for only 50% and 25% of the standard steps, respectively. We quantified the model quality by computing the mean squared prediction error over a set of randomly sampled state-action pairs, which confirmed a monotonic reduction in predictive accuracy across the models.

Table D.6: Effect of Dynamics Model Accuracy on Safety Rates at the time of Inference (Mean over 500 episode rollouts).

Method	**Safe Episodes (%)**	**Episode Reward**	**Mean L2 E**
BC+**V-OCBF** (Known Dynamics)	99.52 ± 0.58	55.91 ± 0.39	0
BC+**V-OCBF** (Learned Dynamics)	98.28 ± 0.54	54.93 ± 0.46	1.48×10
BC+**V-OCBF** (Half-Learned Dynamics)	97.03 ± 0.45	52.44 ± 0.18	2.93×10
BC+**V-OCBF** (Quarter-Learned Dynamics)	96.32 ± 0.73	51.91 ± 0.32	4.46×10

Subsequently, we utilized each model to synthesize the controller and evaluated the resulting empirical safety rates. As detailed in Table D.6, the results indicate that the safety rate experiences only a modest reduction despite the increase in model prediction error. These findings suggest that V-OCBF maintains its effectiveness in mitigating safety violations even when the underlying dynamics model is imperfect, supporting the practical applicability of the method in settings where exact system identification is challenging.

D.4 Additional Experiments on Boat Navigation

To further assess the generalization capabilities of V-OCBF, we evaluated the method on a boat navigation task characterized by nonlinear river drift and static obstacles. This environment, while low-dimensional, presents a control challenge due to the drift dynamics which can actively push the agent into unsafe regions if not properly anticipated. Such drift dynamics are particularly difficult for offline safe RL and CBF-based methods to handle when relying on learned models or indirect constraint formulations.

The system dynamics are given by:

$$(x_1)_{t+1} = (x_1)_t + (a_1 + 2 - 0.5x_2^2) \cdot \Delta t$$
$$(x_2)_{t+1} = (x_2)_t + a_2 \cdot \Delta t$$

where Δt represents discrete time step, a_1, a_2 represent the velocity control action in x_1 and x_2 directions respectively, subject to the actuation limit $a_1^2 + a_2^2 \leq 1$. The term $2 - 0.5x_2^2$ introduces a state-dependent drift along the x_1-axis. The safety constraints are encoded as:

$$\ell(x) := min(\|x - (-0.5, 0.5)^T\| - 0.4, \|x - (-1.0, -1.2)^T\| - 0.4) \tag{D.16}$$

where $\ell(x) < 0$ indicates that the boat is inside a obstacle, thereby ensuring that the sub-level set of $\ell(x)$ defines the failure region.

Table D.7: Boat Navigation Experiment: Percentage Safe Episodes against additional baseline methods. Evaluated over 500 episodes and 5 seed values.

Method	**Safe Episodes (%)**
BC+NCBF [132]	88.43 ± 0.59
CPQ [188]	54.14 ± 0.26
WSAC [183]	79.34 ± 0.87
CDT [101]	69.94 ± 0.42
C2IQL [99]	70.77 ± 0.76
BEAR-Lag [86]	71.69 ± 0.16
COptiDICE [89]	87.88 ± 0.65
FISOR [204]	85.78 ± 0.22
BC+**V-OCBF** (Ours)	**97.56 ± 0.13**

We compared V-OCBF against a broader set of baselines, including CPQ [188], WSAC [183], CDT [101], and C2IQL [99], in addition to the methods evaluated in the main text. The results, summarized in Table D.7, demonstrate that V-OCBF consistently achieves a higher empirical safety rate compared to the baseline methods. Notably, the strongest baseline exhibits a safety rate approximately 9% lower than our approach. These results, combined with the experiments presented in Section 9.5, indicate that V-OCBF provides a robust mechanism for minimizing safety violations across diverse dynamical systems, effectively handling the challenges posed by nonlinear drift and learned dynamics.

D.5 Experimental Details

D.5.1 Experimental Hardware

To keep the evaluation fair and avoid any discriminatory added advantage to any specific experiment, all experiments were conducted on a single system equipped with an 14th Gen Intel Core i9-14900KS CPU, 128GB RAM, and an NVIDIA GeForce RTX 5090 GPU for training and experiment evaluations.

D.5.2 Hyperparameters for the Proposed Algorithm

We have compiled and listed down all the hyperparameters that we used to perform our experiments and report the results. These training settings for all the environments are detailed in the Table D.8. For the MuJoCo environments, we use the widely accepted DSRL [100] dataset. We also use the *Class-K* function $\kappa = \alpha \times B(\cdot)$ where $\alpha = 1$.

D.5.3 Hyperparameters for the Baselines

For the CBF-based baselines (NCBF, iDBF, CCBF), we use the same network architectures and learned dynamics models described above to ensure a fair comparison. Hyperparameters for the remaining safe offline RL baselines (COptiDICE, BEAR-Lag, BC, FISOR) are provided in Table D.9. We use the official implementations of CPQ, CDT, BEAR-Lag and COptiDICE from (OSRL) [100], WSAC from [183], C2IQL from [99] and FISOR from [204], and train all methods on the same DSRL datasets.

Table D.8: Hyperparameters for the Proposed (V-OCBF) Network, and Learned Dynamics Model.

Hyperparameter	**Value**
Network Architecture	Multi-Layer Perceptron (MLP)
Activation Function	ReLU
Optimizer	Adam optimizer
Learning Rate	3×10^{-5}
Discount Factor (γ)	0.99
Autonomous Ground Vehicle Collision Avoidance	
Number of Hidden Layers	2
Hidden Layer Size	256 neurons per layer
Dataset Size	75K
Dynamics Model Hidden Layers	3
Dynamics Model Hidden Layer Size	64
Safe Velocity Hopper	
Number of Hidden Layers	3
Hidden Layer Size	256 neurons per layer
Dataset Size	1.32M
Dynamics Model Hidden Layers	4
Dynamics Model Hidden Layer Size	64
Safe Velocity Half-Cheetah	
Number of Hidden Layers	3
Hidden Layer Size	128 neurons per layer
Dataset Size	249K
Dynamics Model Hidden Layers	4
Dynamics Model Hidden Layer Size	64
Safe Velocity Ant	
Number of Hidden Layers	3
Hidden Layer Size	256 neurons per layer
Dataset Size	2.09M
Dynamics Model Hidden Layers	4
Dynamics Model Hidden Layer Size	64
Safe Velocity Walker2D	
Number of Hidden Layers	3
Hidden Layer Size	256 neurons per layer
Dataset Size	2.12M
Dynamics Model Hidden Layers	4
Dynamics Model Hidden Layer Size	64
Safe Velocity Swimmer	
Number of Hidden Layers	3
Hidden Layer Size	256 neurons per layer
Dataset Size	1.68M
Dynamics Model Hidden Layers	4
Dynamics Model Hidden Layer Size	64

Table D.9: Hyperparameters for all the Baselines network. Optimizer: Adam, LR: 1×10^{-3}, Activation: ReLU (applies to all rows). For COptiDICE and BEAR-Lag refer OSRL, [100], for the official implementations. For FISOR refer [204] for official implementation.

Autonomous Ground Vehicle (AGV)	
BC - Actor	MLPActor, (128,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (128,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (128,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (128,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)
Safe Velocity Hopper	
BC - Actor	MLPActor, (256,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (256,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (256,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (256,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)
Safe Velocity Half-Cheetah	
BC - Actor	MLPActor, (256,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (256,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (256,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (256,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)
Safe Velocity Ant	
BC - Actor	MLPActor, (256,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (256,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (256,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (256,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)
Safe Velocity Walker2D	
BC - Actor	MLPActor, (256,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (256,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (256,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (256,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)
Safe Velocity Swimmer	
BC - Actor	MLPActor, (256,2)
BEAR-Lag - Actor	SquashedGaussianMLPActor, (256,2)
BEAR-Lag - Critic	EnsembleDoubleQCritic, (256,2)
BEAR-Lag - Cost critic	MLP, (256,2)
COptiDICE - Actor (extraction)	MLPActor, (256,2)
COptiDICE - Dual / ν network	DualNet (value-like), (256,2)
FISOR - Actor (diffusion denoiser backbone)	DiffusionDenoiserMLP, (256,2)
FISOR - Feasibility classifier	FeasibilityClassifier, (256,2)

Appendix E

Appendix of Chapter 8

E.1 Proofs

E.1.1 Proof of Theorem 8.4.1

Theorem 8.4.1 (Safety Verification Using Conformal Prediction) Let $\mathcal{S}_\delta$ be the set of states satisfying $\hat{V}_\theta(0, \hat{x}) \leq \delta$, and let $(0, \hat{x}_i)_{i=1,\ldots,N_s}$ be N_s i.i.d. samples from $\mathcal{S}_\delta$. Define α_δ as the safety error rate among these N_s samples for a given δ level. Select a safety violation parameter $\epsilon_s \in (0, 1)$ and a confidence parameter $\beta_s \in (0, 1)$ such that:

$$\sum_{i=0}^{l-1} \binom{N_s}{i} \epsilon_s^i (1 - \epsilon_s)^{N_s - i} \leq \beta_s,$$

where $l = \lfloor (N_s + 1)\alpha_\delta \rfloor$. Then, with the probability of at least $1 - \beta_s$, the following holds:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s.$$

Proof. Before we proceed with the proof of the Theorem (8.4.1), let us look at the following lemma which describes split conformal prediction:
Following the Lemma C.1, we employ a conformal scoring function for safety verification, defined as:

$$s(X) = \hat{V}_{\hat{\pi}_\theta}(0, \hat{x}), \forall \hat{x} \in \mathcal{S}_{\tilde{\delta}},$$

where $\mathcal{S}_\delta$ denotes the set of states satisfying $\hat{V}_\theta(0, \hat{x}) \leq \delta$ and the score function measures the alignment between the induced safe policy and the auxiliary value function.

Next, we sample N_s states from the safe set $\mathcal{S}_\delta$ and compute conformal scores for all sampled states. For a user-defined error rate $\alpha \in [0,1]$, let $\hat{q}$ denote the $\frac{(N_s+1)\alpha}{N_s}$th quantile of the conformal scores. According to [174], the following property holds:

$$\mathop{\mathbb{P}}_{\hat{x}\in\mathcal{S}_{\hat{\delta}}}\left(\hat{V}_{\hat{\pi}_\theta}(\hat{x}_i,0)\leq\hat{q}\right)\sim\text{Beta}(N_s-l+1,l), \tag{E.1}$$

where $l=\lfloor(N_s+1)\alpha\rfloor$.
Define E_s as:

$$E_s:=\mathop{\mathbb{P}}_{\hat{x}\in\mathcal{S}_\delta}\left(\hat{V}_{\hat{\pi}_\theta}(\hat{x}_i,0)\leq\hat{q}\right).$$

Here, E_s is a Beta-distributed random variable. Using properties of cumulative distribution functions (CDF), we assert that $E_s \geq 1-\epsilon_s$ with confidence $1-\beta_s$ if the following condition is satisfied:

$$I_{1-\epsilon_s}(N-l+1,l)\leq\beta_s, \tag{E.2}$$

where $I_x(a,b)$ is the regularized incomplete Beta function and also serves as the CDF of the Beta distribution. It is defined as:

$$I_x(a,b)=\frac{1}{B(a,b)}\int_0^x t^{a-1}(1-t)^{b-1}\,dt,$$

where $B(a,b)$ is the Beta function. From [113](8.17.5), it can be shown that $I_x(n-k,k+1)=\sum_{i=1}^{k}\binom{n}{i}x^i(1-x)^{n-i}$.
Then (E.2) can be rewritten as:

$$\sum_{i=1}^{l-1}\binom{N_s}{i}\epsilon_s^i(1-\epsilon)^{N_s-i}\leq\beta_s, \tag{E.3}$$

Thus, if Equation (E.3) holds, we can say with probability $1-\beta_s$ that:

$$\mathop{\mathbb{P}}_{\hat{x}\in\mathcal{S}_{\hat{\delta}}}\left(\hat{V}_{\hat{\pi}_\theta}(\hat{x}_i,0)\leq\hat{q}\right)\geq 1-\epsilon_s. \tag{E.4}$$

Now, let k denote the number of allowable safety violations. Thus, the safety error rate is given by $\alpha_\delta=\frac{k+1}{N_s+1}$. Let $\hat{q}$ represent the $\frac{(N_s+1)\alpha_\delta}{N_s}$th quantile of the conformal scores. Since k denotes the number of samples for which the conformal score is positive, the $\frac{(N_s+1)\alpha_\delta}{N_s}$th quantile of scores corresponds to the maximum *negative score* amongst the sampled states. This implies that $\hat{q}\leq 0$. From this and Equation (E.4), we can conclude with probability $1-\beta_s$ that:

$$\mathop{\mathbb{P}}_{\hat{x}\in\mathcal{S}_\delta}\left(\hat{V}_{\hat{\pi}_\theta}(0,\hat{x}_i)\leq 0\right)\geq 1-\epsilon_s.$$

From Equation (8.4), it can be inferred that $\forall\ (t, \hat{x})$, $\hat{V}(0, \hat{x}_i) \leq \hat{V}_{\hat{\pi}_\theta}(\hat{x}_i, 0)$. Hence, with probability $1 - \beta_s$, the following holds:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s.$$

□

E.1.2 Proof of Theorem 8.4.2

Theorem 8.4.2 (Performance Quantification Using Conformal Prediction) Suppose $\mathcal{S}^*$ denotes the safe states satisfying $V_\theta(0, x) < \infty$ (or equivalently $\hat{V}_\theta(0, \hat{x}^*) < \delta$) and $(0, x_i)_{i=1,\ldots,N_p}$ are N_p i.i.d. samples from $\mathcal{S}^*$. For a user-specified level α_p, let ψ be the $\frac{\lceil (N_p+1)(1-\alpha_p) \rceil}{N_p} th$ quantile of the scores $(p_i := \frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}})_{i=1,\ldots,N_p}$ on the N_p state samples. Select a violation parameter $\epsilon_p \in (0, 1)$ and a confidence parameter $\beta_p \in (0, 1)$ such that:

$$\sum_{i=0}^{l-1} \binom{N_p}{i} \epsilon_p^i (1 - \epsilon_p)^{N_p - i} \leq \beta_p$$

where, $l = \lfloor (N_p + 1)\alpha_p \rfloor$. Then, the following holds, with probability $1 - \beta_p$:

$$\mathop{\mathbb{P}}_{x \in \mathcal{S}^*} \left(\frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}} \leq \psi \right) \geq 1 - \epsilon_p.$$

where C_{max} is a normalizing factor and denotes the maximum possible cost that could be incurred for any $x \in \mathcal{S}^*$.

Proof. To quantify the performance loss, we employ a conformal scoring function defined as:

$$p(x) := \frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}}, \forall x \in \mathcal{S}^*$$

where the score function measures the alignment between the induced optimal policy and the value function.
Next, we sample N_p states from the state space $\mathcal{S}^*$ and compute conformal scores for all sampled states. For a user-defined error rate $\alpha_p \in [0, 1]$, let ψ denote the $\frac{(N_p+1)\alpha_p}{N_p}$ quantile of the conformal scores. According to [174], the following property holds:

$$\mathop{\mathbb{P}}_{x \in \mathcal{S}^*} \left(\frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}} \leq \psi \right) \sim \text{Beta}(N_p - l + 1, l),$$

where $l = \lfloor (N_p + 1)\alpha_p \rfloor$.
Define E_p as:

$$E_p := \mathop{\mathbb{P}}_{x \in \mathcal{S}^*} \left(\frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}} \leq \psi \right).$$

Here, E_p is a Beta-distributed random variable. Using properties of CDF, we assert that $E_p \geq 1 - \epsilon_p$ with confidence $1 - \beta_p$ if the following condition is satisfied:

$$I_{1-\epsilon_p}(N_p - l + 1, l) \leq \beta_p, \tag{E.5}$$

where $I_x(a, b)$ is the regularized incomplete Beta function. From [113](8.17.5), it can be shown that $I_x(n-k, k+1) = \sum_{i=1}^{k} \binom{n}{i} x^i (1-x)^{n-i}$. Hence, Equation (E.5) can be equivalently stated as:

$$\sum_{i=1}^{l-1} \binom{N_p}{i} \epsilon_p^i (1 - \epsilon_p)^{N_p - i} \leq \beta_p \tag{E.6}$$

Thus, if Equation (E.6) holds, we can conclude with probability $1 - \beta_p$ that:

$$\mathop{\mathbb{P}}_{x \in \mathcal{S}^*} \left(\frac{|V_\theta(0, x_i) - V_{\pi_\theta}(0, x_i)|}{C_{max}} \leq \psi \right) \geq 1 - \epsilon_p.$$

□

E.1.3 Proof of Theorem 8.5.1

Theorem 8.5.1 (Safety Verification Under Bounded Disturbances) Consider a system subject to bounded disturbances $d \in \mathcal{D}$, with the induced rollout cost $\hat{V}_{\hat{\pi}_\theta}$ evaluated under both the learned control policy $\hat{\pi}_\theta$ and the learned worst-case disturbance policy $\hat{d}_\theta$, as defined in (8.31). Let $\mathcal{S}_\delta$ be the set of states satisfying $\hat{V}_\theta(0, \hat{x}) \leq \delta$, and let $(0, \hat{x}_i)_{i=1,\ldots,N_s}$ be N_s i.i.d. samples from $\mathcal{S}_\delta$. Define α_δ as the safety error rate among these N_s samples for a given δ level. Select a safety violation parameter $\epsilon_s \in (0, 1)$ and a confidence parameter $\beta_s \in (0, 1)$ such that:

$$\sum_{i=0}^{l-1} \binom{N_s}{i} \epsilon_s^i (1 - \epsilon_s)^{N_s - i} \leq \beta_s,$$

where $l = \lfloor (N_s + 1)\alpha_\delta \rfloor$. Then, with probability at least $1 - \beta_s$, the following holds:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s.$$

Proof. The proof follows the same conformal prediction framework as the proof of Theorem 8.4.1, but now operates under dynamics subject to bounded disturbances $d \in \mathcal{D}$.

In the robust setting, the learned auxiliary value function $\hat{V}_\theta$ induces both a control policy $\hat{\pi}_\theta$ and a disturbance policy $\hat{d}_\theta$ via Equations (8.29) and (8.30), respectively. The rollout cost is computed under both these policies simultaneously, as defined in Equation (8.31):

$$\hat{V}_{\hat{\pi}_\theta}(t, \hat{x}) = \max \Big\{ C(t, x(t), \mathbf{u}, \mathbf{d}) - z, \max_{s \in [t,T]} g(x(s)) \Big\} \Big|_{\substack{\mathbf{u}=\hat{\pi}_\theta, \\ \mathbf{d}=\hat{d}_\theta}}.$$

We employ the disturbance policy $\hat{d}_\theta$ derived from the learned value function as the adversarial disturbance during rollouts. We note that $\hat{d}_\theta$ is the worst-case disturbance with respect to the *learned* value function, and may not coincide with the true worst-case disturbance. Exhaustively searching over all possible disturbance sequences (*e.g.*, $|\mathcal{D}|^T$ sequences for T time steps) is computationally infeasible. Therefore, the conformal guarantee provided here is with respect to the *learned policy's value function* $\hat{V}_{\hat{\pi}_\theta}$ rather than the true value function.

Following Lemma C.1, we define the conformal scoring function for safety verification as:

$$s(\hat{x}) = \hat{V}_{\hat{\pi}_\theta}(0, \hat{x}), \quad \forall \hat{x} \in \mathcal{S}_\delta.$$

We sample N_s i.i.d. states from $\mathcal{S}_\delta$ and compute conformal scores for all sampled states. For a user-defined error rate $\alpha \in [0, 1]$, let $\hat{q}$ denote the $\frac{(N_s+1)\alpha}{N_s}$th quantile of the conformal scores. By [174], the following holds:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq \hat{q} \right) \sim \text{Beta}(N_s - l + 1, l),$$

where $l = \lfloor (N_s + 1)\alpha \rfloor$.

Defining $E_s := \mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq \hat{q} \right)$ and proceeding identically to the proof of Theorem 8.4.1 (via the regularized incomplete Beta function and its binomial representation), we obtain:

$$\sum_{i=0}^{l-1} \binom{N_s}{i} \epsilon_s^i (1 - \epsilon_s)^{N_s - i} \leq \beta_s \implies \mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq \hat{q} \right) \geq 1 - \epsilon_s,$$

with probability at least $1 - \beta_s$.

Setting $\alpha_\delta = \frac{k+1}{N_s+1}$, where k is the number of states for which the conformal score

is positive, ensures $\hat{q} \leq 0$ (as in the non-robust case). Consequently:

$$\mathop{\mathbb{P}}_{\hat{x} \in \mathcal{S}_\delta} \left(\hat{V}_{\hat{\pi}_\theta}(0, \hat{x}_i) \leq 0 \right) \geq 1 - \epsilon_s.$$

□

Remark E.1.1 (Recovery of non-robust guarantees). When the disturbance set reduces to the singleton $\mathcal{D} = \{0\}$, the disturbance policy $\hat{d}_\theta$ is trivially $d \equiv 0$, the rollout cost $\hat{V}_{\hat{\pi}_\theta}$ coincides with the non-robust version, and Theorem 8.5.1 reduces exactly to Theorem 8.4.1. Thus, the robust safety guarantee subsumes the deterministic one as a special case.

Remark E.1.2 (Nature of the robust guarantee). Unlike the deterministic setting where the conformal guarantee holds against the true system dynamics, the robust guarantee is with respect to the learned worst-case disturbance $\hat{d}_\theta$. While this yields a weaker theoretical guarantee, in practice, the learned disturbance closely approximates the true worst case due to the min-max optimization in training. As demonstrated in the case studies (Section 8.5), these guarantees remain highly effective: when evaluated under random disturbance realizations, the empirical safety violation rates consistently fall well below the certified bounds.

E.1.4 Proof of Theorem 8.5.2

Theorem 8.5.2 (Performance Quantification Under Bounded Disturbances) Consider a system subject to bounded disturbances $d \in \mathcal{D}$, where the induced value function $V_{\pi_\theta}(0,x)$ is obtained by rolling out the learned policy π_θ (from (8.36)) against the learned worst-case disturbance $\hat{d}_\theta$ (from (8.30)). Suppose $\mathcal{S}^*$ denotes the safe states satisfying $V_\theta(0,x) < \infty$ (or equivalently $\hat{V}_\theta(0,\hat{x}^*) < \delta$) and $(0,x_i)_{i=1,\dots,N_p}$ are N_p i.i.d. samples from $\mathcal{S}^*$. For a user-specified level α_p, let ψ be the $\frac{\lceil (N_p+1)(1-\alpha_p)\rceil}{N_p}$th quantile of the scores $(p_i := \frac{|V_\theta(0,x_i)-V_{\pi_\theta}(0,x_i)|}{C_{max}})_{i=1,\dots,N_p}$ on the N_p state samples. Select a violation parameter $\epsilon_p \in (0,1)$ and a confidence parameter $\beta_p \in (0,1)$ such that:

$$\sum_{i=0}^{l-1} \binom{N_p}{i} \epsilon_p^i (1-\epsilon_p)^{N_p-i} \leq \beta_p$$

where, $l = \lfloor (N_p+1)\alpha_p \rfloor$. Then, the following holds, with probability $1-\beta_p$:

$$\mathop{\mathbb{P}}_{x\in\mathcal{S}^*}\left(\frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}} \leq \psi\right) \geq 1-\epsilon_p.$$

where C_{max} is a normalizing factor and denotes the maximum possible cost that could be incurred for any $x \in \mathcal{S}^*$.

Proof. The proof mirrors that of Theorem 8.4.2, adapted to account for the presence of disturbances.

In the robust setting, the induced value function $V_{\pi_\theta}(0,x)$ is obtained by rolling out the system under the learned policy π_θ (derived from the robust auxiliary value function via Equation (8.36)) against the learned worst-case disturbance policy $\hat{d}_\theta$ from Equation (8.30).

We define the conformal scoring function for performance quantification as:

$$p(x) := \frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}}, \quad \forall x \in \mathcal{S}^*,$$

where the score function measures the discrepancy between the learned value function and the value realized under the induced policy against the learned disturbance.

We sample N_p i.i.d. states from $\mathcal{S}^*$ and compute conformal scores. For a user-defined error rate $\alpha_p \in [0,1]$, let ψ denote the $\frac{(N_p+1)\alpha_p}{N_p}$th quantile. By [174]:

$$\mathop{\mathbb{P}}_{x\in\mathcal{S}^*}\left(\frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}} \leq \psi\right) \sim \text{Beta}(N_p - l + 1, l),$$

where $l = \lfloor (N_p + 1)\alpha_p \rfloor$.
Defining $E_p := \mathop{\mathbb{P}}_{x \in \mathcal{S}^*}\left(\frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}} \leq \psi\right)$ and proceeding identically to the proof of Theorem 8.4.2 via the regularized incomplete Beta function:

$$I_{1-\epsilon_p}(N_p - l + 1, l) \leq \beta_p,$$

which, using the binomial representation from [113](8.17.5), gives:

$$\sum_{i=0}^{l-1} \binom{N_p}{i} \epsilon_p^i (1-\epsilon_p)^{N_p - i} \leq \beta_p.$$

Thus, with probability $1 - \beta_p$:

$$\mathop{\mathbb{P}}_{x \in \mathcal{S}^*}\left(\frac{|V_\theta(0,x_i) - V_{\pi_\theta}(0,x_i)|}{C_{max}} \leq \psi\right) \geq 1 - \epsilon_p.$$

□

Remark E.1.3 (Recovery of non-robust performance guarantees). When $\mathcal{D} = \{0\}$, the disturbance vanishes, V_{π_θ} reduces to the non-robust induced value, and Theorem 8.5.2 reduces exactly to Theorem 8.4.2.

Remark E.1.4 (Interpretation of the robust performance bound). The performance bound ψ quantifies the worst-case discrepancy between the learned value function and the rollout cost under the learned disturbance policy. Since $\hat{d}_\theta$ is derived from the trained value function (rather than being the true worst-case adversary), the bound characterizes performance degradation against the *learned adversary*. In practice, as demonstrated in Section 8.5, the learned adversary provides a strong proxy for the true worst case, and the empirical performance degradation under random disturbances remains well within the certified bound ψ.

E.2 Additional Details the systems in the experiments

In this section, we will provide more details about the systems we have used in the experiments section 8.4.5.

E.2.1 Efficient and Safe Boat Navigation

The states, x of the 2D Boat system are $x = [x_1, x_2]^T$, where, x_1, x_2 are the x and y coordinates of the boat respectively. We define the step cost at each step, $l(t, x)$, as the distance from the goal, given by:

$$l(t, x) := \|x - (1.5, 0)^T\|$$

The cost function $C(t, x(t))$ is defined as:

$$C(t, x(t), \mathbf{u}) = \int_t^T l(t, x(t))\, dt \; + \phi(x(T)) \tag{E.7}$$

where T is the time horizon ($2s$ in our experiment), $l(t, x(t)) = ||x(t) - (1.5, 0)^T||$ represents the running cost, and $\phi(x(T)) = ||x(T) - (1.5, 0)^T||$ is the terminal cost. Minimizing this cost drives the boat toward the island.

Consequently, the (augmented) dynamics of the 2D Boat system are:

$$\begin{aligned} \dot{x_1} &= u_1 + 2 - 0.5x_2^2 \\ \dot{x_2} &= u_2 \\ \dot{z} &= -l(t, x) \end{aligned}$$

where u_1, u_2 represents the velocity control in x_1 and x_2 directions respectively, with $u_1^2 + u_2^2 \leq 1$ and $2 - 0.5x_2^2$ specifies the current drift along the x_1-axis.

The safety constraints are formulated as:

$$g(x) := max(0.4 - \|x - (-0.5, 0.5)^T\|, 0.5 - \|x - (-1.0, -1.2)^T\|)) \tag{E.8}$$

where $g(x) > 0$ indicates that the boat is inside a boulder, thereby ensuring that the super-level set of $g(x)$ defines the failure region.

Ground Truth Comparison

We compute the Ground Truth value function using the Level-Set Toolbox [107] and use it as a benchmark in our comparative analysis. To facilitate demonstration, unsafe states are assigned a high value of 20 instead of ∞. The value function in this problem ranges from 0 to 14.76.

As illustrated in Figure E.1, the value function obtained using our method closely approximates the ground truth value function. The unsafe region (highlighted in

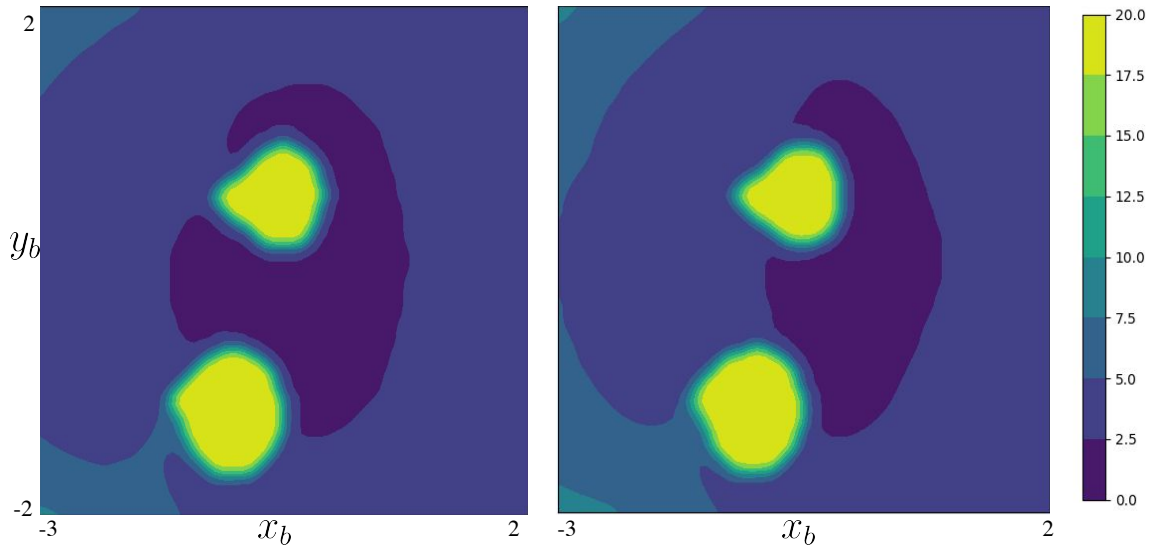


Figure E.1: Heatmap of the value function for the ground truth (left) and our method (right). The yellow region represents the unsafe area. Our method successfully captures most of the safe set, indicating that it is not overly conservative while completely recovering the unsafe regions.

yellow) remains identical in both cases, confirming the safety of the learned value function. The mean squared error (MSE) between the two value functions is 0.36, which is relatively low given the broad range of possible values.

It is also worth mentioning that computing a high-fidelity ground truth value function on a $210 \times 210 \times 210$ grid using the Level Set Toolbox requires approximately 390 minutes. In contrast, our proposed approach learns the value function in 122 minutes, achieving a substantial speedup. This demonstrates that even for systems with a relatively low-dimensional state space, our method efficiently recovers an accurate value function significantly faster than grid-based solvers.

Empirical Validation of the α-β-ϵ relationship and calculation of safety levels

We conducted an experiment to empirically validate the theoretical relationship between α, β, and ϵ. The results are presented in Figure E.2. The figure visualizes the relationship between theoretical and empirical safety metrics across varying levels of δ, and includes the following elements:

- *Safety error rate* (α, purple line) as a function of different δ levels. Computed on the calibration dataset as $\alpha = \frac{k+1}{N_s+1}$, where k is the number of allowable safety violations and N_s is the number of calibration samples.

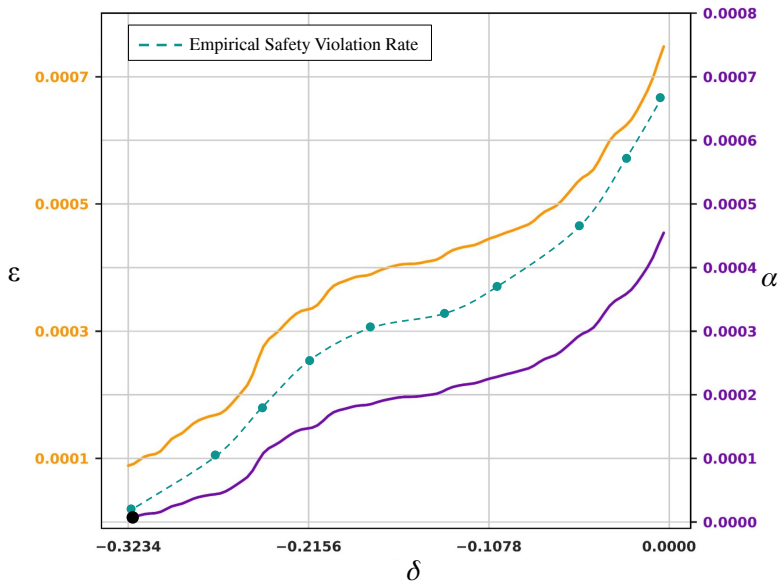


Figure E.2: This figure presents a comparative analysis of the relationships between ϵ–δ, α–δ, and empirical safety violation rate–δ. As observed, the empirical safety violation consistently remains below the theoretical bound, thereby supporting our theoretical guarantees. As ϵ decreases, the corresponding δ approaches zero, indicating that the learned value function incurs negligible safety violations.

- *Theoretical safety violation probability* (ϵ, orange line) as a function of δ. Derived using the theoretical relation in Equation (14).
- *Empirical safety violation probability* (green points) as a function of δ. Computed by sampling 3M initial states from the δ-sublevel set of the learned value function, simulating rollouts, and measuring the observed safety violation rate. This serves as a practical estimate of system safety.

For this experiment, we set $N = 300k$ and $\beta = 10^{-10}$. As shown in the figure, the empirical violation rate remains consistently below the theoretical bound (ϵ) across all values of δ. This demonstrates that our method provides conservative and valid safety guarantees, confirming the soundness of the theoretical relationship in practice.

Additionally, from the δ vs ϵ plot, we can observe that the δ level approaches 0 as the ϵ values approach the chosen safety level of 0.001. Hence, we say that the sub-level set of the auxiliary value function, $\hat{V}(t, \hat{x})$ is safe with a probability of $1 - 0.001 = 0.999$.

E.2.2 Pursuer vehicle tracking an evader

The state, x of a ground vehicle (pursuer) tracking a moving evader is $x = [x_p, y_p, v, \Theta, x_e, y_e, v_{xe}, v_{ye}]^T$ where, x_e, y_e, v, Θ are position, linear velocity and orientation of the pursuer respectively, x_e, y_e, v_{xe}, v_{ye} are the position and the linear velocities of the evader respectively.

We define the step cost at each step, $l(t, x)$, as the distance from the goal, given by:

$$l(t, x) := \|(x_p(t), y_p(t))^T - (x_e(t), y_e(t))^T\|$$

and the terminal cost is $\phi(x(T)) = ||(x_p(T), y_p(T))^T - (x_e(T), y_e(T))^T||$. The cost function $C(t, x(t))$ is defined as:

$$C(t, x(t), \mathbf{u}) = \int_t^T l(t, x(t))\, dt \; + \phi(x(T)) \tag{E.9}$$

where T is the time horizon ($1s$ in this experiment). Minimizing this cost aims to drive the pursuer toward the evader. Consequently, the (augmented) dynamics of the system is as follows:

$$\begin{aligned} &\dot{x_p} = v\cos(\Theta), \quad \dot{y_p} = v\sin(\Theta), \quad \dot{v} = u_1, \quad \dot{\Theta} = u_2, \\ &\dot{x_e} = v_{xe}, \quad \dot{y_e} = v_{ye}, \quad \dot{v_{xe}} = 0, \quad \dot{v_{ye}} = 0, \quad \dot{z} = -l(t, x), \end{aligned}$$

where u_1 represents the linear acceleration control and u_2 represents angular velocity control.

The safety constraints are defined as:

$$\begin{aligned} g(x) :=& max(0.2 - \|x - (0.5, 0.5)^T\|, 0.2 - \|x - (-0.5, 0.5)^T\|, 0.2 - \|x - (-0.5, -0.5)^T\|, \\ & 0.2 - \|x - (0.5, -0.5)^T\|, 0.2 - \|x - (0.0, 0.0)^T\|,)) \end{aligned}$$

which represents 5 obstacles of radius 0.2 units each.

E.2.3 Multi-Agent Navigation

A multi-agent setting with 5 agents. The state of each agent i is represented by $x_i = [x_{a_i}, y_{a_i}, x_{g_i}, y_{g_i}]$, tries to reach its goal while avoiding collisions with others. (x_{a_i}, y_{a_i}) denote the position of the ith agent, while (x_{g_i}, y_{g_i}) represent the goal locations for that agent. We define the step cost at each step, $l(t, x(t))$, as the mean distance of each agent from its respective goal, given by:

$$l(t, x(t)) := \frac{\sum_{i=1}^{5} \|(x_{a_i}(t), y_{a_i}(t)^T - (x_{g_i}(t), y_{g_i}(t))^T\|}{5}$$

The cost function $C(t, x(t), \mathbf{u})$ is defined as:

$$C(t, x(t), \mathbf{u}) := \int_t^T l(t, x(t))\, dt \; + \phi(x(T)) \tag{E.10}$$

where T is the time horizon ($2s$ in this experiment). Minimizing this cost aims to drive each agent towards its goal. Consequently, the (augmented) dynamics of the system is as follows:

$$\begin{aligned}
\dot{x}_{a_i} &= u_{1i}, \forall i \in \{1,2,3,4,5\} \\
\dot{y}_{a_i} &= u_{2i}, \forall i \in \{1,2,3,4,5\} \\
\dot{x}_{g_i} &= 0, \forall i \in \{1,2,3,4,5\} \\
\dot{y}_{g_i} &= 0, \forall i \in \{1,2,3,4,5\} \\
\dot{z} &= -l(t,x)
\end{aligned}$$

where u_{1i}, u_{2i} represents the linear velocity control of each agent i. The safety constraints are defined as:

$$g(x(t)) := \max_{i,j=\{1,..,5\}, i \neq j} (R - \|(x_{a_i}, y_{a_i})^T - (x_{a_j}, y_{a_j})^T\|) \tag{E.11}$$

Comparison of Multi-Agent Navigation with baselines

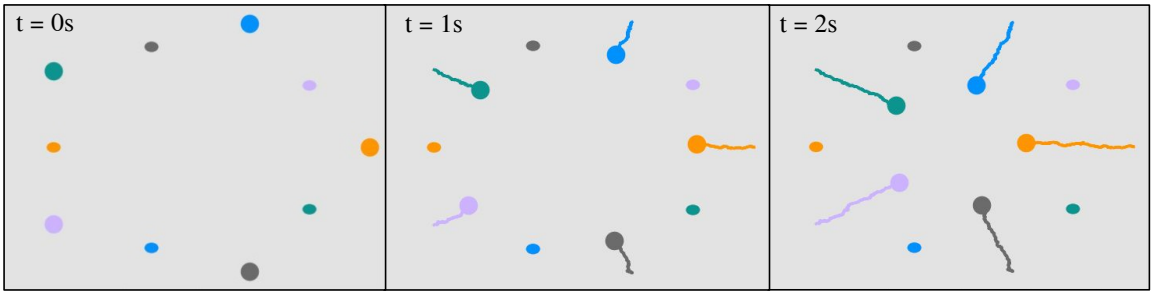


Figure E.3: Snapshots of multi-agent navigation trajectories at different time instances using **MPPI**. The trajectories indicate that the agents adopt a **highly conservative strategy** to prevent collisions. Consequently, this leads to a **reduction in performance**, as the agents **end up very far from their respective goals**.

Figures E.3, E.4, and E.5 illustrate the trajectories obtained by the baseline methods for the Multi-Agent Navigation problem. It can be observed that the trajectories obtained by MPPI and MPPI-SF are highly conservative, implying that these methods prioritize safety to mitigate potential conflicts among agents. In contrast, the policy derived from SAC-Lag fails to maintain safety, resulting in agent collisions. This indicates that as system complexity increases, the baseline methods tend to prioritize either safety or performance, leading to suboptimal behavior and safety violations. Conversely, the proposed approach effectively co-optimizes safety and performance, even in complex high-dimensional settings, achieving superior performance while ensuring safety. The visualization of the trajectories can be found on the project

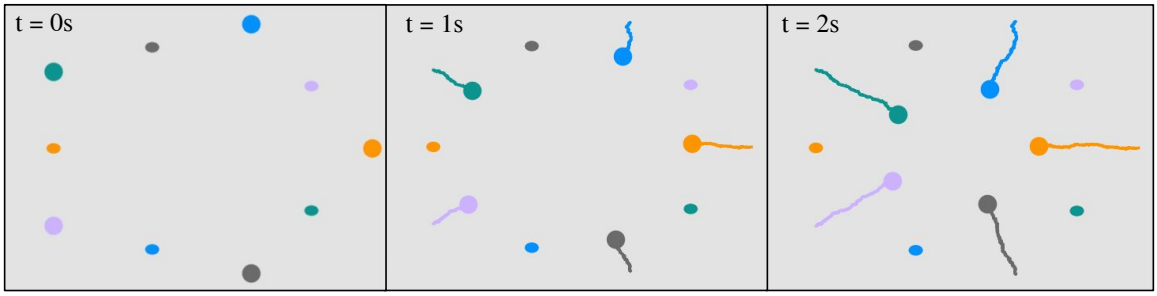


Figure E.4: Snapshots of multi-agent navigation trajectories at different time instances using **MPPI-NCBF**. The observed trajectories demonstrate **suboptimal behavior similar to that of the MPPI policy**. Consequently, this results in high-performance costs, indicating its **inability to effectively co-optimize safety and performance.**

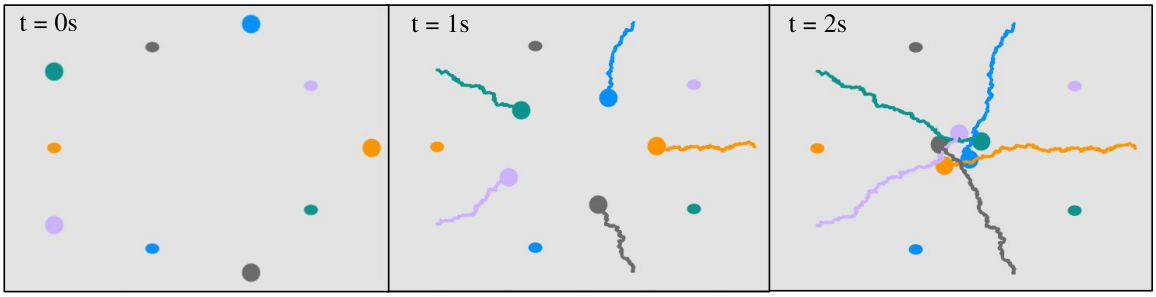


Figure E.5: Snapshots of multi-agent navigation trajectories at different time instances using SAC-Lag. The trajectories indicate that agents **demonstrate less conservative behavior compared to MPPI and MPPI-NCBF, but they lead to collisions**. These **safety violations are critical** and cannot be disregarded, further **highlighting the limitations of the baseline methods in simultaneously optimizing safety and performance.**

website[1].

E.3 Implementation Details of the Algorithms

This section provides an in-depth overview of our algorithm and baseline implementations, including hyperparameter configurations and the cost/reward functions used in the baselines across all experiments.

E.3.1 Experimentation Hardware

All experiments were conducted on a system equipped with an 11th Gen Intel Core i9-11900K @ 3.50GHz × 16 CPU, 128GB RAM, and an NVIDIA GeForce RTX 4090

[1] https://tayalmanan28.github.io/piml-soc/

GPU for training.

E.3.2 Hyperparameters for the Proposed Algorithm

We maintained training settings across all experiments, as detailed below:

Hyperparameter	**Value**
Network Architecture	Multi-Layer Perceptron (MLP)
Number of Hidden Layers	3
Activation Function	Sine function
Hidden Layer Size	256 neurons per layer
Optimizer	Adam optimizer
Learning Rate	2×10^{-5}
Boat Navigation	.
Number of Training Points	65000
Number of Pre Training Epochs	50K
No. of Training Epochs	200K
Pursuer Vehicle Tracking Evader	.
Number of Training Points	65000
Number of Pre Training Epochs	60K
No. of Training Epochs	300K
Multi Agent Navigation	.
Number of Training Points	65000
Number of Pre Training Epochs	60K
No. of Training Epochs	400K

Table E.1: Hyperparameters for the proposed algorithm

E.3.3 MPPI based baselines

For all the experiments we consider the MPPI cost term as follows:

$$C_{MPPI} = C(t, x(t), \mathbf{u}) + \lambda \max(g(x), 0) \tag{E.12}$$

where, λ is the trade-off parameter, $C(t, x(t), \mathbf{u})$, $g(x)$ are the cost functions and safety functions as defined in Appendix E.2. Following is the list of hyperparameters we have used for MPPI experiments in all the cases:

E.3.4 SAC-Lag hyperparameters

For all the experiments, we consider the reward term as follows:

Hyperparameter	Value
Trade-off parameter (λ)	100
Planning Horizon	20
Softmax Lambda	200
No. of Rollouts	8000

Table E.2: Hyperparameters for MPPI Baselines

$$R_{SAC-Lag} = -C(t, x(t), \mathbf{u}) - \mathbb{I}_{g(x)>0} \times (100) + \mathbb{I}_{l(t,x(t))<0.1} \times (100) \tag{E.13}$$

where, $C(t, x(t), \mathbf{u})$, $g(x)$ are the cost functions and safety functions as defined in Appendix E.2. Table E.3 provides the list of hyperparameters we have used for SAC experiments in all the cases.

Parameter	Value
Policy Architecture	Multi-Layer Perceptron (MLP)
learning rate	3×10^{-4}
buffer size	$1,000,000$
learning starts	$10,000$
batch size	256
Target network update rate (τ)	0.005
Discount factor (γ)	0.99
Boat Navigation	.
Number of Training Steps	1,000,000
Pursuer Vehicle Tracking Evader	.
Number of Training Steps	2,500,000
Multi Agent Navigation	.
Number of Training Steps	1,000,000

Table E.3: General Hyperparameters of SAC in our experiments

E.3.5 PPO-Lag hyperparameters

For all the experiments, we consider the reward term as follows:

$$R_{PPO-Lag} = -C(t, x(t), \mathbf{u}) - \mathbb{I}_{g(x)>0} \times (100) + \mathbb{I}_{l(t,x(t))<0.1} \times (100) \tag{E.14}$$

where, $C(t, x(t), \mathbf{u})$, $g(x)$ are the cost functions and safety functions as defined in Appendix E.2. Table E.4 provides the list of hyperparameters we have used for PPO experiments in all the cases.

Parameter	Value
Policy Architecture	Multi-Layer Perceptron (MLP)
learning rate	3×10^{-4}
buffer size	$1,000,000$
learning starts	$10,000$
batch size	256
Target network update rate (τ)	0.005
Discount factor (γ)	0.99
Boat Navigation	.
Number of Training Steps	1,000,000
Pursuer Vehicle Tracking Evader	.
Number of Training Steps	2,500,000
Multi Agent Navigation	.
Number of Training Steps	1,000,000

Table E.4: General Hyperparameters of PPO in our experiments

E.3.6 CPO hyperparameters

For all the experiments, we consider the reward term as follows:

$$R_{CPO} = -C(t, x(t), \mathbf{u}) + \mathbb{I}_{l(t,x(t))<0.1} \times (100) \tag{E.15}$$

where $C(t, x(t), \mathbf{u})$, $g(x)$ are the cost functions and safety functions as defined in Appendix E.2. For the CPO implementation, we have used the training settings used in [28]. Table E.5 provides the list of hyperparameters we have used for CPO experiments in all the cases.

Parameter	Value
Policy Architecture	Multi-Layer Perceptron (MLP)
Batch Size	128
Target KL Divergence	0.01
Entropy Coefficient	0.0
Reward Discount Factor (γ)	0.99
Cost Discount Factor (γ_c)	0.99
GAE Lambda (λ)	0.95
Cost GAE Lambda (λ_c)	0.95
Critic Norm Coefficient	0.001
Penalty Coefficient	0.0
Conjugate Gradient Damping	0.1
Conjugate Gradient Iterations	15
Actor Hidden Sizes	$[256, 256]$
Critic Hidden Sizes	$[256, 256]$
Critic Learning Rate	0.001
Boat Navigation	.
Number of Training Steps	10,000,000
Pursuer Vehicle Tracking Evader	.
Number of Training Steps	25,000,000
Multi Agent Navigation	.
Number of Training Steps	10,000,000

Table E.5: CPO Hyperparameters from OmniSafe Configuration used for our experiments

Bibliography

[1] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. "Fossil: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks". In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 2021, pp. 1–11.

[2] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. "Formal synthesis of Lyapunov neural networks". In: *IEEE Control Systems Letters* 5.3 (2020), pp. 773–778.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. "Constrained policy optimization". In: *International conference on machine learning*. PMLR. 2017, pp. 22–31.

[4] Albert Altarovici, Olivier Bokanowski, and Hasnaa Zidani. "A general Hamilton-Jacobi framework for non-linear state-constrained control problems". In: *ESAIM: Control, Optimisation and Calculus of Variations* 19.2 (2013), pp. 337–357.

[5] E. Altman. *Constrained Markov Decision Processes*. Stochastic Modeling Series. Taylor & Francis, 1999. ISBN: 9780849303821. URL: https://books.google.co.in/books?id=3X9S1NM2iOgC.

[6] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. "Control barrier functions: Theory and applications". In: *18th European control conference (ECC)*. IEEE. 2019, pp. 3420–3431.

[7] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pp. 6271–6278.

[8] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876. DOI: 10.1109/tac.2016.2638961.

[9] Mahathi Anand and Majid Zamani. "Formally verified neural network control barrier certificates for unknown systems". In: *IFAC-PapersOnLine* 56.2 (2023), pp. 2431–2436.

[10] Anastasios N. Angelopoulos and Stephen Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. 2022. arXiv: 2107.07511 [cs.LG]. URL: https://arxiv.org/abs/2107.07511.

[11] Leemon Baird. "Residual algorithms: Reinforcement learning with function approximation". In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37.

[12] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. "Hamilton-jacobi reachability: A brief overview and recent advances". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 2242–2253.

[13] Somil Bansal and Claire J Tomlin. "Deepreach: A deep learning approach to high-dimensional reachability". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 1817–1824.

[14] Andrew J Barry, Anirudha Majumdar, and Russ Tedrake. "Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 484–490.

[15] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: A Survey". In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43.

[16] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. "Reciprocal n-body collision avoidance". In: *Robotics Research*. Springer, 2011, pp. 3–19.

[17] Jur van den Berg, Ming Lin, and Dinesh Manocha. "Reciprocal velocity obstacles for real-time multi-agent navigation". In: *2008 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2008, pp. 1928–1935.

[18] Bitcraze. *Crazyflie Python Library*. https://github.com/bitcraze/crazyflie-lib-python. Accessed: 2018-10-16. 2018.

[19] Franco Blanchini. "Set invariance in control". In: *Automatica* 35.11 (1999), pp. 1747–1767.

[20] Javier Borquez, Kaustav Chakraborty, Hao Wang, and Somil Bansal. "On Safety and Liveness Filtering Using Hamilton–Jacobi Reachability Analysis". In: *IEEE Transactions on Robotics* 40 (2024), pp. 4235–4251. DOI: 10.1109/TRO.2024.3454470.

[21] Urs Borrmann, Li Wang, Aaron D. Ames, and Magnus Egerstedt. "Control barrier certificates for safe swarm behavior". In: *IFAC-PapersOnLine*. Vol. 48. 27. 2015, pp. 68–73.

[22] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[23] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. "RT-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[24] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. "Safe learning in robotics: From learning-based control to safe reinforcement learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2022), pp. 411–444.

[25] Fernando Castañeda, Haruki Nishimura, Rowan Thomas McAllister, Koushil Sreenath, and Adrien Gaidon. "In-Distribution Barrier Functions: Self-Supervised Policy Filters that Avoid Out-of-Distribution States". In: *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. Ed. by Nikolai Matni, Manfred Morari, and George J. Pappas. Vol. 211. Proceedings of Machine Learning Research. PMLR, 2023, pp. 286–299. URL: https://proceedings.mlr.press/v211/castaneda23a.html.

[26] A. Chakravarthy and D. Ghose. "Obstacle avoidance in a dynamic environment: a collision cone approach". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.5 (1998), pp. 562–574. DOI: 10.1109/3468.709600.

[27] Animesh Chakravarthy and Debasish Ghose. "Generalization of the collision cone approach for motion safety in 3-D environments". In: *Autonomous Robots* 32.3 (2012), pp. 243–266. ISSN: 1573-7527. DOI: 10.1007/s10514-011-9270-z.

[28] Jingrun Chen, Rui Du, and Keke Wu. "A comparison study of deep Galerkin method and deep Ritz method for elliptic problems with different boundary conditions". In: *arXiv preprint arXiv:2005.04554* (2020).

[29] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. "Decision transformer: Reinforcement learning via sequence modeling". In: *Advances in neural information processing systems* 34 (2021), pp. 15084–15097.

[30] Mo Chen, Sylvia L. Herbert, Mahesh S. Vashishtha, Somil Bansal, and Claire J. Tomlin. "Decomposition of reachable sets and tubes for a class of nonlinear systems". In: *IEEE Transactions on Automatic Control*. Vol. 63. 11. 2018, pp. 3675–3688.

[31] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel 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. 01. 2019, pp. 3387–3395.

[32] Jason J. Choi, Jasmine Jerry Aloor, Jingqi Li, Maria G. Mendoza, Hamsa Balakrishnan, and Claire J. Tomlin. "Resolving Conflicting Constraints in Multi-Agent Reinforcement Learning with Layered Safety". In: *Proceedings of Robotics: Science and Systems*. Los Angeles, USA, 2025. URL: https://arxiv.org/abs/2505.02293.

[33] Jason J. Choi, Donggun Lee, Koushil Sreenath, Claire J. Tomlin, and Sylvia L. Herbert. "Robust Control Barrier–Value Functions for Safety-Critical Control". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. Austin, TX, USA: IEEE Press, 2021, pp. 6814–6821. DOI: 10.1109/CDC45484.2021.9683085. URL: https://doi.org/10.1109/CDC45484.2021.9683085.

[34] Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. "Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton–Jacobi equations arising from optimal control and differential games problems". In: *Journal of Scientific Computing* 73 (2017), pp. 617–643.

[35] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. "Risk-constrained reinforcement learning with percentile risk criteria". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6070–6120.

[36] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. "Lyapunov-based safe policy optimization for continuous control". In: *arXiv preprint arXiv:1901.10031* (2019).

[37] Andrew Clark. "Control barrier functions for stochastic systems". In: *Automatica* 130 (2021), p. 109688.

[38] Andrew Clark. "Verification and synthesis of control barrier functions". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 6105–6112.

[39] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2019.

[40] Michael G. Crandall and Pierre-Louis Lions. "Viscosity solutions of Hamilton–Jacobi equations". In: *Transactions of the American Mathematical Society* 277.1 (1983), pp. 1–42.

[41] Hongkai Dai, Chuanrui Jiang, Hongchao Zhang, and Andrew Clark. "Verification and Synthesis of Compatible Control Lyapunov and Control Barrier Functions". In: *arXiv preprint arXiv:2406.18914* (2024).

[42] Hongkai Dai and Frank Permenter. "Convex synthesis and verification of control-Lyapunov and barrier functions with input constraints". In: *2023 American Control Conference (ACC)*. IEEE. 2023, pp. 4116–4123.

[43] Jérôme Darbon and Stanley Osher. "Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere". In: *Research in the Mathematical Sciences* 3.1 (2016), p. 19.

[44] Charles Dawson, Sicun Gao, and Chuchu Fan. "Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control". In: *IEEE Transactions on Robotics* (2023).

[45] Charles Dawson, Bethany Lowenkamp, Dylan Goff, and Chuchu Fan. "Learning safe, generalizable perception-based hybrid control with certificates". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1904–1911.

[46] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. "Safe nonlinear control using robust neural Lyapunov-barrier functions". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1724–1735.

[47] Sarah Dean, Andrew J. Taylor, Ryan K. Cosner, Benjamin Recht, and Aaron D. Ames. "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions". In: *Conference on Robot Learning (CoRL)*. 2020, pp. 654–670.

[48] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. "Goal-conditioned Imitation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/c8d3a760ebab631565f8509d84b3b3f1-Paper.pdf.

[49] Anish Dixit, Lars Lindemann, Skylar X. Wei, Matthew Cleaveland, George J. Pappas, and Joel W. Burdick. "Adaptive conformal prediction for motion planning among dynamic agents". In: *Learning for Dynamics and Control Conference (L4DC)*. 2023.

[50] Lester E Dubins. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents". In: *American Journal of Mathematics* 79.3 (1957), pp. 497–516.

[51] Alec Edwards, Andrea Peruffo, and Alessandro Abate. "Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models". In: *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. 2024, pp. 1–10.

[52] Lawrence C. Evans and Panagiotis E. Souganidis. “Differential Games and Representation Formulas for Solutions of Hamilton–Jacobi–Isaacs Equations”. In: *Indiana University Mathematics Journal* 33.5 (1984), pp. 773–797.

[53] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. “Efficient and accurate estimation of Lipschitz constants for deep neural networks”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[54] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using velocity obstacles”. In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[55] Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. “Reach-avoid problems with time-varying dynamics, targets and constraints”. In: *Proceedings of the 18th international conference on hybrid systems: computation and control*. 2015, pp. 11–20.

[56] Jaime F. Fisac, Neil F. Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J. Tomlin. “Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8550–8556. DOI: 10.1109/ICRA.2019.8794107.

[57] Carl Folkestad, Skylar X. Wei, and Joel W. Burdick. “KoopNet: Joint Learning of Koopman Bilinear Models and Function Dictionaries with Application to Quadrotor Trajectory Tracking”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 1344–1350. DOI: 10.1109/ICRA46639.2022.9811896.

[58] Carlos E. García, David M. Prett, and Manfred Morari. “Model predictive control: Theory and practice—A survey”. In: *Automatica* 25.3 (1989), pp. 335–348. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(89)90002-2. URL: https://www.sciencedirect.com/science/article/pii/0005109889900022.

[59] Javier Garcıa and Fernando Fernández. “A comprehensive survey on safe reinforcement learning”. In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[60] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Dario Amodei, Timothy Mann, and Pushmeet Kohli. “On the effectiveness of interval bound propagation for training verifiably robust models”. In: *arXiv preprint arXiv:1810.12715*. 2018.

[61] Lars Grüne, Jürgen Pannek, Lars Grüne, and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

[62] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. "Safe multi-agent reinforcement learning for multi-robot control". In: *Artificial Intelligence*. Vol. 319. 2023, p. 103905.

[63] David R Ha and Jürgen Schmidhuber. "World Models". In: *ArXiv* abs/1803.10122 (2018). URL: https://api.semanticscholar.org/CorpusID:4807711.

[64] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

[65] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to control: Learning behaviors by latent imagination". In: *International Conference on Learning Representations (ICLR)*. 2020.

[66] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. "Mastering diverse domains through world models". In: *arXiv preprint arXiv:2301.04104* (2023).

[67] Suiyi He, Jun Zeng, Bike Zhang, and Koushil Sreenath. *Rule-Based Safety-Critical Control Design using Control Barrier Functions with Application to Autonomous Lane Change*. 2021. DOI: 10.48550/ARXIV.2103.12382.

[68] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing". In: *2007 American Control Conference*. 2007, pp. 2296–2301. DOI: 10.1109/ACC.2007.4282788.

[69] Kai-Chieh Hsu, Haimin Hu, and Jaime F. Fisac. "The Safety Filter: A Unified View of Safety-Critical Control in Autonomous Systems". In: *Annual Review of Control, Robotics, and Autonomous Systems* 7.Volume 7, 2024 (2024), pp. 47–72. ISSN: 2573-5144. DOI: https://doi.org/10.1146/annurev-control-071723-102940. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-control-071723-102940.

[70] Shao-Chen Hsu, Xiangru Xu, and Aaron D. Ames. "Control barrier function based quadratic programs with application to bipedal robotic walking". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 4542–4548.

[71] Hanjiang Hu, Yujie Yang, Tianhao Wei, and Changliu Liu. "Verification of neural control barrier functions with symbolic derivative bounds propagation". In: *arXiv preprint arXiv:2410.16281* (2024).

[72] Pushpak Jagtap, George J Pappas, and Majid Zamani. "Control barrier functions for unknown nonlinear systems using Gaussian processes". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 3699–3704.

[73] Pushpak Jagtap, Sadegh Soudjani, and Majid Zamani. "Formal synthesis of stochastic systems via control barrier certificates". In: *IEEE Transactions on Automatic Control* 66.7 (2020), pp. 3097–3110.

[74] Pushpak Jagtap, Abdalla Swikir, and Majid Zamani. "Compositional construction of control barrier functions for interconnected control systems". In: *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. 2020, pp. 1–11.

[75] Mrdjan Jankovic. "Robust Control Barrier Functions for Constrained Stabilization of Nonlinear Systems". In: *Automatica* 96 (2018), pp. 359–367.

[76] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. "Planning with Diffusion for Flexible Behavior Synthesis". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9902–9915.

[77] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. "Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark". In: *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2023. URL: https://openreview.net/forum?id=WZmlxIuIGR.

[78] 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". In: *Journal of Machine Learning Research* 25.285 (2024), pp. 1–6. URL: http://jmlr.org/papers/v25/23-0681.html.

[79] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. "Neural certificates for safe control policies". In: *arXiv preprint arXiv:2006.08465* (2020).

[80] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. "Scalable deep reinforcement learning for vision-based robotic manipulation". In: *Conference on Robot Learning (CoRL)*. 2018, pp. 651–673.

[81] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*. Vol. 113. Springer Science & Business Media, 1991.

[82] Hassan K. Khalil. *Nonlinear Systems*. 3rd. Upper Saddle River, NJ: Prentice Hall, 2002.

[83] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *The International Journal of Robotics Research* 5.1 (1986), pp. 90–98.

[84] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[85] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. “Offline reinforcement learning with implicit q-learning”. In: *International Conference on Learning Representations* (2022).

[86] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. “Stabilizing off-policy q-learning via bootstrapping error reduction”. In: *Advances in neural information processing systems* 32 (2019).

[87] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. “Conservative Q-Learning for Offline Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

[88] Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. “Optidice: Offline policy optimization via stationary distribution correction estimation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6120–6130.

[89] Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. “COptiDICE: Offline Constrained Reinforcement Learning via Stationary Distribution Correction Estimation”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=FLA55mBee6Q.

[90] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. “End-to-end training of deep visuomotor policies”. In: *Journal of Machine Learning Research*. Vol. 17. 39. 2016, pp. 1–40.

[91] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[92] Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. “When Data Geometry Meets Deep Function: Generalizing Offline Reinforcement Learning”. In: *The Eleventh International Conference on Learning Representations*. 2023.

[93] Zhichao Li. *Comparison between safety methods control barrier function vs. reachability analysis*. 2021. DOI: 10.48550/ARXIV.2106.13176. URL: https://arxiv.org/abs/2106.13176.

[94] Albert Lin and Somil Bansal. "Generating Formal Safety Assurances for High-Dimensional Reachability". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10525–10531. DOI: 10.1109/ICRA48891.2023.10160600.

[95] Albert Lin and Somil Bansal. "Verification of neural reachable tubes via scenario optimization and conformal prediction". In: *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*. Ed. by Alessandro Abate, Mark Cannon, Kostas Margellos, and Antonis Papachristodoulou. Vol. 242. Proceedings of Machine Learning Research. PMLR, 2024, pp. 719–731. URL: https://proceedings.mlr.press/v242/lin24a.html.

[96] Qian Lin, Bo Tang, Zifan Wu, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. "Safe Offline Reinforcement Learning with Real-Time Budget Constraints". In: *International Conference on Machine Learning*. 2023.

[97] Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J. Pappas. "Safe planning in dynamic environments using conformal prediction". In: *IEEE Robotics and Automation Letters*. Vol. 8. 8. 2023, pp. 5116–5123.

[98] Simin Liu, Changliu Liu, and John Dolan. "Safe control under input limits with neural control barrier functions". In: *Conference on Robot Learning*. PMLR. 2023, pp. 1970–1980.

[99] Zifan LIU, Xinran Li, and Jun Zhang. "C2IQL: Constraint-Conditioned Implicit Q-learning for Safe Offline Reinforcement Learning". In: *Forty-second International Conference on Machine Learning*. 2025. URL: https://openreview.net/forum?id=97N3XNtFwy.

[100] Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, and Ding Zhao. "Datasets and Benchmarks for Offline Safe Reinforcement Learning". In: *Journal of Data-centric Machine Learning Research* (2024).

[101] Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. "Constrained decision transformer for offline safe reinforcement learning". In: *International Conference on Machine Learning*. 2023.

[102] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[103] Michael Lutter, Christian Ritter, and Jan Peters. "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning". In: *International Conference on Learning Representations*. 2019.

[104] John Lygeros. "On reachability and minimum cost optimal control". In: *Automatica* 40.6 (2004), pp. 917–927.

[105] Liying Ma and Khashayar Khorasani. "Constructive feedforward neural networks using Hermite polynomial activation functions". In: *IEEE Transactions on Neural Networks* 16.4 (2005), pp. 821–833.

[106] Jiří Matoušek and Bernd Gärtner. *Understanding and using linear programming*. Vol. 1. Springer, 2007.

[107] I. Mitchell. "A toolbox of level set methods". In: *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS.pdf* (2004).

[108] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on automatic control* 50.7 (2005), pp. 947–957.

[109] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. "Human-Level Control through Deep Reinforcement Learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[110] Tamas G Molnar, Ryan K Cosner, Andrew W Singletary, Wyatt Ubellacker, and Aaron D Ames. "Model-free safety-critical control for robotic systems". In: *IEEE robotics and automation letters* 7.2 (2021), pp. 944–951.

[111] Ameneh Nejati, Abolfazl Lavaei, Pushpak Jagtap, Sadegh Soudjani, and Majid Zamani. "Formal verification of unknown discrete-and continuous-time systems: A data-driven approach". In: *IEEE Transactions on Automatic Control* 68.5 (2023), pp. 3011–3024.

[112] Quan Nguyen and Koushil Sreenath. "Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints". In: *2016 American Control Conference (ACC)*. 2016, pp. 322–328. DOI: 10.1109/ACC.2016.7524935.

[113] F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and eds. M. A. McClain. *NIST Digital Library of Mathematical Functions*. Release 1.1.11 of 2023-09-15. National Institute of Standards and Technology, 2023. URL: https://dlmf.nist.gov/.

[114] Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. "Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[115] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. "Inductive confidence machines for regression". In: *European Conference on Machine Learning (ECML)*. Springer. 2002, pp. 345–356.

[116] Pablo A Parrilo. "Semidefinite programming relaxations for semialgebraic problems". In: *Mathematical programming* 96 (2003), pp. 293–320.

[117] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. "Training robust neural networks using Lipschitz bounds". In: *IEEE Control Systems Letters* 6 (2021), pp. 121–126.

[118] Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. "Automated and Formal Synthesis of Neural Barrier Certificates for Dynamical Models". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer International Publishing, 2021, pp. 370–388.

[119] Lerrel Pinto and Abhinav Gupta. "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3406–3413.

[120] Philip Polack, Florent Altché, Brigitte d'Andréa-Novel, and Arnaud de La Fortelle. "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.

[121] Stephen Prajna, Ali Jadbabaie, and George J Pappas. "A framework for worst-case and stochastic safety verification using barrier certificates". In: *IEEE Transactions on Automatic Control* 52.8 (2007), pp. 1415–1428.

[122] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons, 1994.

[123] Zengyi Qin, Dawei Sun, and Chuchu Fan. "Sablas: Learning safe control for black-box dynamical systems". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1928–1935.

[124] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. "Learning safe multi-agent control with decentralized neural barrier certificates". In: *International Conference on Learning Representations (ICLR)*. 2021.

[125] Zhizhen Qin, Tsui-Wei Weng, and Sicun Gao. "Quantifying safety of learning-based self-driving control using almost-barrier functions". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 12903–12910.

[126] Quan Quan. *Introduction to multicopter design and control*. Springer, 2017.

[127] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[128] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. *Physics Informed Deep Learning (Part I & II): Data-driven Solutions of Nonlinear Partial Differential Equations*. 2017. arXiv: 1711.10561 [cs.AI].

[129] Rajesh Rajamani. *Vehicle dynamics and control*. en. 2nd ed. Mechanical Engineering Series. New York, NY: Springer, Mar. 2014.

[130] Alex Ray, Joshua Achiam, and Dario Amodei. *Benchmarking Safe Exploration in Deep Reinforcement Learning*. 2019. URL: https://cdn.openai.com/safexp-short.pdf.

[131] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*. Vol. 293. Springer Science & Business Media, 2013.

[132] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V. Dimarogonas, Stephen Tu, and Nikolai Matni. "Learning Control Barrier Functions from Expert Demonstrations". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 3717–3724. DOI: 10.1109/CDC42340.2020.9303785.

[133] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. pearson, 2016.

[134] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. "A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 9835–9846.

[135] E. Schmerling. "hj_reachability: Hamilton-Jacobi reachability analysis in JAX". In: *https://github.com/StanfordASL/hj_reachability* (2021).

[136] Michael Schneeberger, Florian Dörfler, and Silvia Mastellone. "SOS construction of compatible control Lyapunov and barrier functions". In: *IFAC-PapersOnLine* 56.2 (2023), pp. 10428–10434.

[137] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[138] Glenn Shafer and Vladimir Vovk. "A tutorial on conformal prediction". In: *Journal of Machine Learning Research* 9 (2008), pp. 371–421.

[139] Aditya Singh, Zeyuan Feng, and Somil Bansal. "Exact Imposition of Safety Boundary Conditions in Neural Reachable Tubes". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025. URL: https://arxiv.org/abs/2404.00814.

[140] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. "Implicit Neural Representations with Periodic Activation Functions". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 7462–7473.

[141] Oswin So, Andrew Clark, and Chuchu Fan. "Almost-sure safety guarantees of stochastic zero-control barrier functions do not hold". In: *arXiv preprint arXiv:2312.02430* (2023).

[142] Oswin So and Chuchu Fan. "Solving Stabilize-Avoid Optimal Control via Epigraph Form and Deep Reinforcement Learning". In: *Robotics: Science and Systems*. 2023.

[143] Oswin So, Cheng Ge, and Chuchu Fan. "Solving Minimum-Cost Reach Avoid using Reinforcement Learning". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=jzngdJQ2lY.

[144] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems". In: *arXiv preprint arXiv:2310.15478* (2023).

[145] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 11532–11539.

[146] Halil Mete Soner. "Optimal Control with State-Space Constraint I". In: *SIAM Journal on Control and Optimization* 24.3 (1986), pp. 552–561. DOI: 10.1137/0324032. eprint: https://doi.org/10.1137/0324032. URL: https://doi.org/10.1137/0324032.

[147] Adam Stooke, Joshua Achiam, and Pieter Abbeel. "Responsive Safety in Reinforcement Learning by PID Lagrangian Methods". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9133–9143. URL: https://proceedings.mlr.press/v119/stooke20a.html.

[148] Lucas Streichenberg, Elia Trevisan, Jen Jen Chung, Roland Siegwart, and Javier Alonso-Mora. "Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 1379–1385. DOI: 10.1109/ICRA48891.2023.10161511.

[149] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[150] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 12. 1999.

[151] Ihab Tabbara and Hussein Sibai. *Learning Neural Control Barrier Functions from Offline Data with Conservatism*. 2025. arXiv: 2505.00908 [cs.LG]. URL: https://arxiv.org/abs/2505.00908.

[152] Xiao Tan, Wenceslao Shaw Cortez, and Dimos V. Dimarogonas. "High-Order Barrier Functions: Robustness, Safety, and Performance-Critical Control". In: *IEEE Transactions on Automatic Control* 67.6 (2022), pp. 3021–3028. DOI: 10.1109/TAC.2021.3089639.

[153] Manan Tayal, Bhavya Giri Goswami, Karthik Rajgopal, Rajpal Singh, M. Tejas Rao, Jishnu Keshavan, Pushpak Jagtap, and Shishir Nadubettu Yadukumar. "A Collision Cone Approach for Control Barrier Functions". In: *IEEE Transactions on Control Systems Technology* (2026), pp. 1–14. DOI: 10.1109/TCST.2026.3661160.

[154] Manan Tayal, Bhavya Giri Goswami, Karthik Rajgopal, Pushpak Jagtap, and Shishir Kolathaya. "Collision Cone Control Barrier Functions: Experimental Validation on UGVs for Kinematic Obstacle Avoidance". In: *2024 American*

Control Conference (ACC). 2024, pp. 325–331. DOI: 10.23919/ACC60939.2024.10644338.

[155] Manan Tayal and Shishir Kolathaya. "Polygonal Cone Control Barrier Functions (PolyC2BF) for safe navigation in cluttered environments". In: *2024 European Control Conference (ECC)*. IEEE. 2024, pp. 2212–2217.

[156] Manan Tayal and Shishir Kolathaya. "Safe Legged Locomotion using Collision Cone Control Barrier Functions (C3BFs)". In: *arXiv preprint arXiv:2309.01898* (2023).

[157] Manan Tayal and Akshay Nambi. "ShieldVLA: Feasibility-Aware Safety Alignment for Vision-Language-Action Models". In: *arXiv preprint arXiv:2609.13231* (2026).

[158] Manan Tayal, Aditya Singh, Pushpak Jagtap, and Shishir Kolathaya. "Cp-ncbf: A conformal prediction-based approach to synthesize verified neural control barrier functions". In: *arXiv preprint arXiv:2503.17395* (2025).

[159] Manan Tayal, Aditya Singh, Pushpak Jagtap, and Shishir Kolathaya. "Semi-Supervised Safe Visuomotor Policy Synthesis using Barrier Certificates". In: *2025 IEEE 64th Conference on Decision and Control (CDC)*. 2025.

[160] Manan Tayal, Aditya Singh, Shishir Kolathaya, and Somil Bansal. "A Physics-Informed Machine Learning Framework for Safe and Optimal Control of Autonomous Systems". In: *Forty-second International Conference on Machine Learning*. 2025. URL: https://openreview.net/forum?id=SrfwiloGQF.

[161] Manan Tayal, Aditya Singh, Shishir Kolathaya, and Somil Bansal. "MADPINN: A Decentralized Physics-Informed Machine Learning Framework for Safe and Optimal Multi-Agent Control". In: *Proceedings of the AAAI Workshop on Multi-Agent Path Finding* (2026).

[162] Manan Tayal, Aditya Singh, Ganga Nair, Shishir Kolathaya, and Somil Bansal. "A Robust Physics-Informed Machine Learning Framework for Safe and Optimal Control of Autonomous Systems". In: *Submitted to The International Journal of Robotics Research (IJRR)* (2026). Under review. URL: https://drive.google.com/drive/folders/1F8chiiyOM1asnKAeXV3oo6t8oCtk1c4S?usp=sharing.

[163] Manan Tayal, Rajpal Singh, Jishnu Keshavan, and Shishir Kolathaya. "Control barrier functions in dynamic uavs for kinematic obstacle avoidance: A collision cone approach". In: *2024 American Control Conference (ACC)*. IEEE. 2024, pp. 3722–3727.

[164] Manan Tayal, Mumuksh Tayal, and Ravi Prakash. "EpiFlow: Bridging Safety and Performance in Offline Reinforcement Learning via Epigraph-Guided Flow Matching". In: *Accepted for Presentation at advances in neural information processing systems (NeurIPS), 2026. Also accepted at ICLR 2026 Workshop: VerifAI-2: The Second Workshop on AI Verification in the Wild*. 2026. URL: https://openreview.net/forum?id=U3EW3m3Ys5.

[165] Manan Tayal, Hongchao Zhang, Pushpak Jagtap, Andrew Clark, and Shishir Kolathaya. "Learning a Formally Verified Control Barrier Function in Stochastic Environment". In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 4098–4104. DOI: 10.1109/CDC56724.2024.10886052.

[166] Mumuksh Tayal, Manan Tayal, and Ravi Prakash. "Bridging Safety and Performance in Autonomous Systems using Offline Reinforcement Learning". In: *ICML 2026 Workshop on Agents in the Wild: Safety, Security, and Beyond*. 2026. URL: https://openreview.net/forum?id=ffX7tGwAD8.

[167] Mumuksh Tayal, Manan Tayal, and Ravi Prakash. "Safe Flow Q-Learning: Offline Safe Reinforcement Learning with Reachability-Based Flow Policies". In: *Reinforcement Learning Journal* 7 (2026).

[168] Mumuksh Tayal, Manan Tayal, Aditya Singh, Shishir Kolathaya, and Ravi Prakash. "V-OCBF: Learning Safety Filters from Offline Data via Value-Guided Offline Control Barrier Functions". In: *Transactions on Machine Learning Research* (2026). ISSN: 2835-8856. URL: https://openreview.net/forum?id=PGO9mpIyyb.

[169] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. "Reward Constrained Policy Optimization". In: *International Conference on Learning Representations*. 2018.

[170] Phani Thontepu, Bhavya Giri Goswami, Manan Tayal, Neelaksh Singh, Shyam Sundar P I, Shyam Sundar M G, Suresh Sundaram, Vaibhav Katewa, and Shishir Kolathaya. "Collision Cone Control Barrier Functions for Kinematic Obstacle Avoidance in UGVs". In: *2023 Ninth Indian Control Conference (ICC)*. 2023, pp. 293–298. DOI: 10.1109/ICC61519.2023.10442173.

[171] Vincent Tjeng, Kai Xiao, and Russ Tedrake. "Evaluating robustness of neural networks: An extreme value theory approach". In: *International Conference on Learning Representations (ICLR)*. 2019.

[172] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[173] Mukun Tong, Charles Dawson, and Chuchu Fan. "Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 10511–10517.

[174] Vladimir Vovk. *Conditional validity of inductive conformal predictors*. 2012. arXiv: 1209.2673 [cs.LG]. URL: https://arxiv.org/abs/1209.2673.

[175] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. New York: Springer, 2005.

[176] Kim P. Wabersich, Andrew J. Taylor, Jason J. Choi, Koushil Sreenath, Claire J. Tomlin, Aaron D. Ames, and Melanie N. Zeilinger. "Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems". In: *IEEE Control Systems Magazine* 43.5 (2023), pp. 137–177. DOI: 10.1109/MCS.2023.3291885.

[177] Chuanzheng Wang, Yiming Meng, Stephen L Smith, and Jun Liu. "Safety-critical control of stochastic systems using stochastic control barrier functions". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 5924–5931.

[178] Hao Wang, Adityaya Dhande, and Somil Bansal. "Cooptimizing Safety and Performance With a Control-Constrained Formulation". In: *IEEE Control Systems Letters* 8 (2024), pp. 2739–2744. DOI: 10.1109/LCSYS.2024.3511429.

[179] Li Wang, Aaron D. Ames, and Magnus Egerstedt. "Safety barrier certificates for collisions-free multirobot systems". In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674.

[180] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. "Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification". In: *Advances in Neural Information Processing Systems* 34 (2021).

[181] Sifan Wang, Yujun Teng, and Paris Perdikaris. "Understanding and mitigating gradient flow pathologies in physics-informed neural networks". In: *SIAM Journal on Scientific Computing* 43.5 (2021), A3055–A3081.

[182] Christopher J. C. H. Watkins and Peter Dayan. "Q-Learning". In: *Machine Learning* 8.3–4 (1992), pp. 279–292.

[183] Honghao Wei, Xiyue Peng, Arnob Ghosh, and Xin Liu. "Adversarially Trained Weighted Actor-Critic for Safe Offline Reinforcement Learning". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=82Ndsr4OS6.

[184] Tyler Westenbroek, Ayush Agrawal, Fernando Castaneda, S Shankar Sastry, and Koushil Sreenath. "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers". In: *IFAC-PapersOnLine* 54.5 (2021), pp. 19–24.

[185] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving". In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1603–1622. DOI: 10.1109/TRO.2018.2865891.

[186] Wei Xiao and Calin Belta. "Control barrier functions for systems with high relative degree". In: *2019 IEEE 58th conference on decision and control (CDC)*. IEEE. 2019, pp. 474–479.

[187] Wei Xiao and Calin Belta. "High-Order Control Barrier Functions". In: *IEEE Transactions on Automatic Control* 67.7 (2022), pp. 3655–3662. DOI: 10.1109/TAC.2021.3105491.

[188] Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. "Constraints penalized q-learning for safe offline reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 8753–8760.

[189] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. "Automatic perturbation analysis for scalable certified robustness and beyond". In: *Advances in Neural Information Processing Systems* 33 (2020).

[190] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. "Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=nVZtXBI6LNn.

[191] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. "Robustness of Control Barrier Functions for Safety Critical Control." In: *IFAC-PapersOnLine* 48.27 (2015). Analysis and Design of Hybrid Systems ADHS, pp. 54–61. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.152.

[192] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. "Mean field multi-agent reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 5571–5580.

[193] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Baez, Curtis Lott, Mac Schwager, and Yi Wu. "The surprising effectiveness of PPO in cooperative multi-agent games". In: *Advances in Neural Information Processing Systems* 35 (2022).

[194] Xia Zeng, Wang Lin, Zhengfeng Yang, Xin Chen, and Lilei Wang. "Darboux-type barrier certificates for safety verification of nonlinear hybrid systems". In: *Proceedings of the 13th International Conference on Embedded Software*. 2016, pp. 1–10.

[195] Hongchao Zhang, Luyao Niu, Andrew Clark, and Radha Poovendran. "Fault Tolerant Neural Control Barrier Functions for Robotic Systems under Sensor Faults and Attacks". In: *arXiv preprint arXiv:2402.18677* (2024).

[196] Hongchao Zhang, Zhizhen Qin, Sicun Gao, and Andrew Clark. "SEEV: Synthesis with Efficient Exact Verification for ReLU Neural Barrier Functions". In: *Advances in Neural Information Processing Systems* 37 (2025), pp. 101367–101392.

[197] Hongchao Zhang, Manan Tayal, Jackson Cox, Pushpak Jagtap, Shishir Kolathaya, and Andrew Clark. "Stochastic neural control barrier functions". In: *IEEE Transactions on Automatic Control (TAC)* (2026).

[198] Hongchao Zhang, Junlin Wu, Yevgeniy Vorobeychik, and Andrew Clark. "Exact verification of relu neural control barrier functions". In: *Advances in neural information processing systems* 36 (2023), pp. 5685–5705.

[199] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. "A Branch and Bound Framework for Stronger Adversarial Attacks of ReLU Networks". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. 2022, pp. 26591–26604.

[200] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. "Efficient Neural Network Robustness Certification with General Activation Functions". In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 4939–4948. URL: https://arxiv.org/pdf/1811.00866.pdf.

[201] Songyuan Zhang, Oswin So, Mitchell Black, Zachary Serlin, and Chuchu Fan. "Solving Multi-Agent Safe Optimal Control with Distributed Epigraph Form MARL". In: *Proceedings of Robotics: Science and Systems*. 2025.

[202] Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. "Safe reinforcement learning: A survey". In: *arXiv preprint arXiv:2302.03122* (2023).

[203] Weiye Zhao, Tairan He, Tianhao Wei, Simin Liu, and Changliu Liu. "Safety index synthesis via sum-of-squares programming". In: *2023 American Control Conference (ACC)*. IEEE. 2023, pp. 732–737.

[204] Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. "Safe offline reinforcement learning with feasibility-guided diffusion model". In: *International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=cM95sT3gM3.