WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Electrical & Systems Engineering

Dissertation Examination Committee:
Andrew Clark, Chair
Ioannis Kantaros
Bruno Sinopoli
Yevgeniy Vorobeychik
Shen Zeng
Ning Zhang

Resilient Safe Control of Autonomous Systems

by
Hongchao Zhang

A dissertation presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2025
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

Scientists are indeed like backpackers: there is no definitive end, only new beginnings. Ph.D. is my first long journey exploring the world full of wonderful discoveries and innovations. Throughout this journey, many people in my life have given me the support, faith, and confidence that made this accomplishment possible. I am deeply grateful to you all.

First, I would like to express my sincere gratitude to my advisor, Prof. Andrew Clark, for his guidance and support in my Ph.D. He provides me with his invaluable guidance and unwavering encouragement. His insightful feedback has helped me recognize and overcome my weaknesses. More importantly, he offers me freedom to explore the multi-disciplinary areas, including learning-enabled control, robotics, and security of cyber-physical systems.

I want to thank the rest of my dissertation committee: Prof. Bruno Sinopoli, Prof. Yevgeniy Vorobeychik, Prof. Ning Zhang Prof. Ioannis Kantaros and Prof. Shen Zeng, for their insightful suggestions and expert guidance. Their collective expertise and support facilitated my exploration of multi-disciplinary areas. Their research wisdom and insightful perspectives also encouraged me to think outside the box.

All members of the Lab, past and present, have helped in various ways, including Dr. Qiqiang Hou, Dr. Luyao Niu, Dr. Zhouchi Li, Mr. Shiyu Cheng, Mr. Chuanrui Jiang, Mr. Aobo Lyu, and Mr. Jackson Cox. I also want to thank my collaborators, Prof. Sicun Gao, Dr. Hongkai Dai, Prof. Radha Poovendran, Prof. Bhaskar Ramasubramanian, Prof. Pushpak Jagtap, Prof. Shishir Kolathaya, Dr. Junlin Wu, Dr. Jinwen Wang, Dr, Ao Li, Dr. Dinuka Sahabandu, Dr. Zhizhen Qin, and Dr. Manan Tayal. I also want to thank the ESE office, Mr. Angel Algarin, Mr. Aaron Beagle, Dr. Stacia Burd, Ms. Allise Davis, and Ms. Madi Hester.

My gratitude goes to my family for their constant love, support, and patience. You have always been my rock. To my wonderful wife, Zehui Jiang, thank you for your incredible understanding, unwavering support, and endless encouragement. I wish to remember my grandfather Fengming Zhang lovingly.

Finally, I would like to thank our sponsors for their generous support. This work was supported by This work was supported by National Science Foundation grants CNS-1941670,

CMMI-2418806, Office of Naval Research grant N00014-17-1-2946, and Air Force Office of Scientific Research grant FA9550-22-1-0054.

Hongchao Zhang

Washington University in St. Louis
August 2025

ABSTRACT OF THE DISSERTATION

Resilient Safe Control of Autonomous Systems

by

Hongchao Zhang

Doctor of Philosophy in Electrical Engineering

Washington University in St. Louis, 2025

Professor Andrew Clark, Chair

Asimov's Three Laws of Robotics famously outlined fundamental safety principles governing human-robot interaction. This foundational concept of safety is paramount for today's autonomous systems, such as robots, which possess inherent cyber-physical properties. With the increasingly widespread application of autonomous systems in real-world environments, the challenges facing research on formal safety verification have grown even more significant. However, end-to-end verification of such complex, integrated systems remains an open and formidable challenge due to their high dimensionality, nonlinearity, and the use of learning-based components. This thesis approaches this challenge by pursuing verifiably safe autonomy from two complementary directions: (i) safe control of learning-enabled systems providing formal guarantees and (ii) resilient safe control that maintains formal safety guarantees under extreme scenarios such as sensor faults and cyber-physical attacks.

The first half of this dissertation presents the formal verification of autonomous systems that integrate learning-enabled components. It starts with the safety verification of neural control barrier functions (NCBF) employing Rectified Linear Unit (ReLU) activation functions. By leveraging a generalization of Nagumo's theorem, we propose exact safety conditions for deterministic systems. To manage computational complexity, we enhance the efficiency of verification and synthesis using a VNN-based (Verification of Neural Networks) search

algorithm and a neural breadth-first search algorithm. We further propose the synthesis and verification of safe control for stochastic systems.

The second half of this dissertation broadens the scope of end-to-end verification by explicitly accounting for imperfections and perturbations. We first proposed Fault-Tolerant Stochastic CBFs and NCBFs to provide safety guarantees for autonomous systems under state estimation error caused by low-dimensional sensor faults and attacks. We then investigate the unique challenges posed by Light Detection And Ranging (LiDAR) perception attacks. We propose a fault detection, identification, and isolation mechanism for 2D and 3D LiDAR and provide safe control under attacks.

Chapter 1

Introduction

An autonomous system is a system capable of performing tasks and making decisions independently, without continuous human guidance [1]. These systems integrate sensors, actuators, computational units, and software algorithms to perceive their environment, process information, and execute actions to achieve specific objectives. Therefore, they are inherently cyber-physical systems (CPS), characterized by the seamless integration of computational algorithms and physical components. The physical components of autonomous systems include sensors for perception (e.g., Inertial Motion Units, Global Positioning System, LiDAR, cameras, radar), actuators for physical interaction (motors, hydraulic systems), and mechanical structures providing physical integrity and mobility. These components collectively define the dynamical behavior of the autonomous system, typically represented by differential equations capturing the relationships among system states, inputs, and physical properties. The cyber component encompasses state estimation algorithms, control policies, data processing algorithms, and communication networks, collectively responsible for implementing the decision-making and control tasks. Control policies leverage sensory data and state estimations to issue commands to actuators, thus directly influencing the system's physical behavior.

In recent years, autonomous systems, e.g., autonomous vehicles and AI-embodied robots, have been increasingly deployed in real-world, safety-critical scenarios. These systems operate in dynamic and uncertain environments, often in close interaction with human users [2]. To ensure public trust and prevent catastrophic failures, assured safety is paramount [3]. Achieving this requires not only the design of safe control policies but also formal verification of the underlying control and perception modules. However, end-to-end verification of such complex, integrated systems remains an open and formidable challenge due to their high dimensionality, nonlinearity, and the use of learning-based components. This thesis approaches this challenge by pursuing verifiably safe autonomy from two complementary directions: (i) safe control of learning-enabled systems providing formal guarantees, and

(ii) resilient safe control that maintains formal safety guarantees under extreme scenarios such as sensor faults and cyber-physical attacks. In what follows, we first provide a brief overview of the safe control of autonomous systems. We then discuss the challenges of safe learning-enabled systems, followed by a discussion of resilient safe control. Finally, we summarize the contributions of this thesis and outline its structure.

1.1 Safe Control of Autonomous Systems

The increasing deployment of autonomous systems necessitates formally guaranteeing their safe and reliable operation, as any safety violation can lead to catastrophic consequences, including economic losses[4, 5], severe injuries, or loss of human lives [6]. The safety requirements of these systems, with applications including medicine, energy, and robotics [7], have motivated recent research to design safe control policies. Safety requirements can be formulated as the positive invariance of a given safe region, meaning the system remains in the safe region indefinitely [8]. Various approaches for safety-critical control have been proposed, including Hamilton-Jacobi Reachability (HJR) analysis [9, 10, 11], Barrier certificates [12], model predictive control [13], finite-state approximation [14] and Control Barrier Functions (CBFs) [15]. Among those methods, CBFs have the advantage that they can be readily integrated into existing control policies by adding linear constraints on the control input. Due to their ease of implementation and compatibility with various safety and performance criteria [16], CBF-based approaches have been applied in deterministic [8] and stochastic control systems [17, 18, 19]. A CBF maps the system state to a scalar and serves as a constraint in an optimization problem. The constraints ensure that the state remains inside the region where the CBF is nonnegative, which is a subset of a given safe region. More recently, CBFs have emerged as promising approaches to safe control, due to their compatibility with a wide variety of control laws [8, 20, 18, 21, 22, 23, 24]. These optimization-based controllers are usually formulated as Quadratic Programs (QPs), with CBF constraints, known as CBF-QPs. CBF-QPs require known CBFs for the control system in a specific safe region. A CBF-based optimization is proposed for systems with nominal controllers, known as a safety filter. The optimization problem minimizes the norm of the additive control effort such that the overall control input satisfies CBF conditions. Safety filters provide formal guarantees without limitations on nominal control policies.

However, these CBFs are not always given and need to be synthesized. For control systems with polynomial dynamics, CBF synthesis can be formulated as sum-of-squares (SOS) constrained optimization problems [25, 26, 27]. Unfortunately, these SOS-based problems do not easily scale to high-dimensional systems and are not directly applicable to systems with non-polynomial dynamics, including learning-enabled robotic systems [28] and neural network dynamical models [29, 30].

As artificial intelligence (AI) systems increase in size rapidly, acquire new capabilities, and are deployed in safety-critical systems, their safety becomes extremely important. Autonomous systems utilizing neural networks to reason and learn from data are known as learning-enabled systems. Learning-enabled systems show great promise due to the uniform approximability of neural networks; however, inherited from neural networks' black-box nature, formal verification is an open problem. Ensuring system safety requires more than improving accuracy, efficiency, and scalability: it requires the representation of neural networks and generalized safety certification of systems with neural networks integrated.

Safe reinforcement learning (RL) [31, 32, 33] has been proposed to ensure or maximize the probability of remaining safe. However, the lack of guarantees of safe RL is an open problem impeding the deployment of these methods on real-world applications. While control barrier function (CBF)-based safety filters offer formal safety guarantees, polynomial CBFs often struggle to scale to high-dimensional systems. This mismatch creates a fundamental tension: the safety filter enforces provable safety, yet its capability is constrained when paired with a learning-enabled nominal controller that is limited by the expressiveness of polynomial CBFs.

To address these limitations, Neural CBFs (NCBFs) [34, 23, 35] have been proposed to represent CBFs with feed-forward Neural Networks (NNs) by exploiting the uniform approximability of NNs. NCBFs have shown substantial promise in applications including robotic manipulation [23], navigation [36, 37], and flight control [38]. After synthesizing an NCBF, the NCBF must be verified in order to ensure that the resulting CBF-QP is feasible everywhere in the state space, and to ensure that the super-level-set of the NCBF is contained in the safe region [39, 34]. Verification-In-The-Loop (VITL), also known as Counterexample-guided synthesis, has been implemented in neural barrier certificates [40, 41] and NCBF verification [42, 43]. VITL synthesis of NCBFs is dependent on an effective and efficient verifier, which is an open problem due to the scalability issue of verifying a neural network and its derivatives.

1.2 Resilient Safe Control

Autonomous systems rely on perception modules to estimate states, including their own states as well as the environmental states. Perception modules develop this understanding using data from sensors such as cameras, Global Positioning Systems, RADAR, and Light Detection and Ranging (LiDAR) [3].

Sensor faults and malicious attacks provide inaccurate, arbitrary readings. These inaccurate measurements bias estimates of the system state, leading to erroneous control signals that drive the true system state to an unsafe operating point. Sensor faults and attacks can cause arbitrary errors in the sensor measurements and system dynamics, which is challenging for existing CBF-based approaches such as [44] that assume that noises and disturbances are either bounded or come from a known probability distribution.

Countermeasures such as Fault-Tolerant Control (FTC) [45, 46] have been proposed to accommodate faults, attacks and failures. Existing FTC approaches focus on maintaining performance and do not provide provable safety guarantees. Countermeasures incorporating disturbance observer-based CBF are proposed to ensure robust safety of systems with model uncertainties [47, 48] and model-free safe reinforcement learning [49]. With the growing attention on faults and attacks, safety guarantees on systems under faulty components or adversarial environments have become an active research area.

Modeling and detection of sensor faults and attacks have been extensively studied [50, 51, 52]. Secure system state estimation using measurements from proprioceptive sensors has been investigated in [53, 54]. Closed-loop safety-critical control under sensor faults and attacks has been recently studied in [55, 56]. However, these approaches are applicable to CPS using only proprioceptive sensors. When exteroceptive sensors such as LiDAR are adopted by CPS, the impact of attacks on the output of the nonlinear filters used to process LiDAR measurements are not incorporated into the aforementioned safety-critical control designs [55, 56], rendering them less effective.

LiDARs, which measure the distances from the LiDAR transceiver to obstacles, provide a 360° view and a 2D or 3D representation, namely a point cloud, of the environment. Since LiDAR perception has a significant impact on the safety-critical decisions of AVs, many prior research efforts have been made to investigate the security of LiDAR perception. The

LiDAR perception can be compromised by relay attacks [57][58][59] and adversarial object attacks [60][61]. In relay attacks, a relay spoofer injects adversarial points in the point cloud, disturbs the outputs of the object detection algorithms, and either creates the perception of a fake object or hides an existing object. In adversarial object attacks, a well-designed object fools the object detection algorithms and makes itself undetectable. While sensor fusion-based methods can partially mitigate the impact of LiDAR attacks [62], such methods can still be thwarted by an adversary who can target multiple sensor modalities simultaneously.

1.3 Contribution of this Thesis

1.3.1 Neural CBF for Deterministic Systems

The key challenge for this class of NCBFs is that most methodologies for safety verification are based on proving that the derivative of the barrier function is nonnegative at the boundary of the safe region, and hence the barrier function remains nonnegative for all time. Since the ReLU activation function is not continuously differentiable, this approach is inapplicable. We resolve this challenge with the following contributions.

- We derive exact safety conditions for ReLU NCBFs by leveraging a generalization of Nagumo's theorem for proving invariance of sets with nonsmooth boundaries.
- We propose an algorithm to verify that an NCBF satisfies our derived safety conditions. Our approach leverages the piece-wise linearity of ReLU neural networks and decomposes the NCBF into hyperplanes and hinges. In order to mitigate the complexity of this stage, we show that it suffices to consider the boundary of the safe region.
- We propose a VNN-based searching algorithm to identify boundary hyperplanes and hinges by over-approximate the input-output relationship of the NCBF with Interval Bound Propagation and linear relaxation. After decomposing the NCBF, we verify safety by solving a set of nonlinear programs

We identify that the computational bottleneck of NCBF verification is the inherent requirement of verifying each linear segment of the neural network. We mitigate this bottleneck by (i)

developing a training procedure that reduces the number of segments that must be verified and (ii) constructing verification algorithms that efficiently enumerate the linear segments at the safety boundary and exploit easily-checked sufficient conditions to reduce computation time.

- Towards (i), we introduce a regularizer to the loss function that penalizes the dissimilarity of activation patterns along the CBF boundary.
- Towards (ii), we propose a breadth-first search algorithm for efficiently enumerating the boundary segments, as well as tight linear over-approximations of the nonlinear optimization problems for verifying each segment.
- Moreover, we integrate the synthesis and verification components by incorporating safety counterexamples returned by the safety verifier into the training dataset.
- Our simulation results demonstrate significant improvements in verification efficiency and reliability across a range of benchmark systems.

1.3.2 Neural CBF for Stochastic Systems

The synthesis of Stochastic Neural CBFs (SNCBFs) requires additional verification to check if the synthesized SNCBF is both correct and feasible for the given system dynamics and safety constraints. The synthesis of SNCBFs is dependent on an effective and efficient verifier, which is an open problem due to the scalability issue of verifying a neural network and its derivatives. Furthermore, existing work primarily focuses on deterministic systems and leaves stochastic NCBFs less studied.

To investigate this challenge, we propose a training framework to synthesize provably valid NCBFs for continuous-time, stochastic systems. Our methodology establishes completeness guarantees by deriving a validity condition, which ensures efficacy across the entire state space with only a finite number of data points. We train the network robustly by enforcing Lipschitz bounds on the neural network and its Jacobian and Hessian. We make the following contributions towards synthesizing and verifying NCBFs for stochastic systems.

- 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.
- We introduce the Stochastic NCBF (SNCBF) with ReLU activation functions and derive sufficient safety conditions using Tanaka's formula.
- We utilize the piecewise linearity of ReLU NNs, formulate the verification of ReLU SNCBFs as nonlinear programs and propose practical algorithms for efficient verification.
- We frame the Verification-In-The-Loop (VITL) synthesis with efficient verifiers for both smooth and ReLU NCBFs synthesis. The VITL relaxes the dense sampling and single-layer assumption of the smooth SNCBF.
- We validate our approach in three cases, namely, the inverted pendulum, Darboux, and the unicycle model. The experiments illustrate that both smooth and ReLU SNCBFs can output verifiably safe results while covering a larger safe subset compared to the baseline approach of a Fault-Tolerant SNCBF without VITL.

1.3.3 Resilient Safe Control under Low-Dimensional Sensor Faults

Sensor faults and malicious attacks provide inaccurate, arbitrary readings. These inaccurate measurements bias estimates of the system state, leading to erroneous control signals that drive the true system state to an unsafe operating point. Existing FTC approaches focus on maintaining performance and do not provide provable safety guarantees. To fill the blank, we make the following specific contributions:

- We propose High-order Stochastic CBFs (HOSCBF) for the system with high relative degree and propose FT-SCBFs with high order degree to ensure finite time safety when sensor faults occur. We propose an SOS-based scheme to verify the feasibility of constraints of FT-SCBFs with high relative degree.
- We compose HOSCBFs with Control Lyapunov Functions (CLFs) to provide joint guarantees on safety and stability under sensor faults.

- We evaluate our approach via a case study. The proposed HOSCBF-CLF ensures safety and convergence of a wheeled mobile robot (WMR) system in the presence of a sensor attack.

- We propose FT-NCBFs for robotic systems under sensor faults and attacks. We derive the necessary and sufficient conditions for FT-NCBFs to guarantee safety. Based on the derived conditions, we develop a data-driven method to learn FT-NCBFs.

- We develop a fault-tolerant framework which utilizes our proposed FT-NCBFs for safety-critical control synthesis. We prove that the synthesized control inputs guarantee safety under all fault and attack patterns.

- We evaluate our approach using two case studies on the obstacle avoidance problem of a mobile robot and the spacecraft rendezvous problem. We show that our approach guarantees the robot to satisfy the safety constraint regardless of the faults and attacks, whereas the baseline employing the existing NCBFs fails.

1.3.4 Resilient Safe Control under LiDAR Perception Attacks

Sensors have been shown to be vulnerable to faults and malicious attacks, under which ensuring CPS safety becomes more challenging. Existing works are applicable to CPS using only low-dimensional sensors. When high-dimensional sensors such as LiDAR are adopted by CPS, the impact of attacks on the output of the nonlinear filters used to process LiDAR measurements are not incorporated into the aforementioned safety-critical control designs, rendering them less effective. Safety-critical control under LiDAR spoofing attacks is an open problem. To address the problem, we make the following contributions.

- We propose a fault tolerant state estimation algorithm that is resilient to attacks against proprioceptive sensors and 2D-LiDAR measurements. Our approach reconstructs a simulated scan based on a state estimate and a precomputed map of the environment. We leverage this reconstruction to remove false sensor inputs as well as detect and remove spoofed LiDAR measurements.

- We propose a fault tolerant safe control design using control barrier certificates. We present a sum-of-squares program to compute a control barrier certificate, which verifies

a given safety constraint in the presence of estimation errors due to noise and attacks. We prove bounds on the probability that our synthesized control input guarantees safety.

- We validate our proposed framework using a UAV delivery system equipped with multiple sensors including a 2D-LiDAR. We show that the UAV successfully avoids the obstacles when navigating in an urban environment using our synthesized control law, while crashes into the unsafe region using a baseline.

- We propose a safe control system for 3D-LiDAR-perception-based AVs based on the point cloud from neighboring vehicles. In the proposed system, a Fault Detection, Identification, and Isolation (FDII) module detects and classifies the attacks, and updates the unsafe region for the vehicle. A safe controller guarantees the safety of the system based on the updated unsafe region.

- We analyze the correctness of the results from the FDII module. We show that the FDII module can detect and classify attacks correctly, and output the unsafe region containing the projection of the obstacles.

- Our results are validated through CARLA [63], in which we show that the proposed FDII procedure correctly detects multiple attack types and reconstructs the true unsafe region. We then show that, under our control algorithm, the vehicle reaches the given target while avoiding an obstacle.

1.4 Structure of Thesis

The core of this thesis is formed by two interconnected threads that advance the theory and practice of safe autonomy in complex, uncertain environments. The first thread focuses on the verifiable safety of learning-enabled systems, including both deterministic and stochastic dynamics. It develops formal conditions and scalable algorithms for the synthesis and exact verification of neural control barrier functions (NCBFs), leveraging the structure of ReLU networks and integrating verification into the learning process. The second thread addresses resilient safe control under faults and adversarial conditions. It introduces fault-tolerant control frameworks capable of detecting, isolating, and mitigating sensor faults for cyber-physical systems while maintaining provable safety guarantees. These two threads share a

unifying goal: enabling trustworthy autonomy through learning-enabled control methods that are not only expressive but also certifiably safe and robust. The thesis is structured as follows:

Chapter 2 Safety Verification of Deterministic Systems: The content of this chapter is based on the following works. Exact Verification of ReLU Neural Control Barrier Functions is coauthored with Dr. Junlin Wu, Prof. Yevgeniy Vorobeychik and Prof. Andrew Clark. In the paper, we explore novel theoretical advances addressing the inherent difficulties posed by the non-differentiability of ReLU activation functions used in neural networks. Specifically, the paper introduces generalized Nagumo's conditions for invariance in the presence of non-smooth barriers, followed by detailed algorithmic solutions leveraging piecewise-linear decomposition using VNN-based approaches. SEEV: Synthesis with Efficient Exact Verification for ReLU Neural Barrier Functions is coauthored with Dr. Zhizhen Qin, Prof. Sicun Gao and Prof. Andrew Clark. In the paper, we further mitigated the scalability issue. We introduce a regularizer in synthesis to help reduce computational complexity in verification, as well as an efficient exact verification by proposing neural breadth-first search and hierarchical verification. We concludes with comprehensive numerical results that illustrate significant gains in verification efficiency and scalability compared to existing approaches.

Chapter 3 Safety Verification of Stochastic Systems: The content of this chapter is based on the work presented in our submission to IEEE Transactions on Automatic Control entitledStochastic Neural Control Barrier Functions, which is coauthored with Dr. Manan Tayal, Mr. Jackson Cox, Prof. Pushpak Jagtap, Prof. Shishir Kolathaya and Prof. Andrew Clark. Building upon deterministic verification methods, this chapter extends safety guarantees to stochastic autonomous systems, introducing Smooth and ReLU-based Stochastic Neural Control Barrier Functions (SNCBFs). It explores novel theoretical results and synthesis techniques tailored for systems subjected to probabilistic uncertainties and noise. The chapter systematically describes verification and synthesis frameworks, embedding formal safety verification into the training loop of SNCBFs. Numerical experiments, including stochastic vehicle dynamics and robotic navigation, demonstrate the effectiveness and efficiency of the proposed methods, illustrating advances in handling real-world uncertainty in safety-critical contexts.

Chapter 4 Resilient Safe Control under Low-Dimensional Sensor Faults: This chapter addresses safety and resilience in the presence of sensor faults and attacks. Section 4.3-4.4 is based on Safe Control for Nonlinear Systems under Faults and Attacks via Control Barrier Functions, which is coauthored with Dr. Zhouchi Li and Prof. Andrew Clark. In the paper, we investigate safety and resilience in the presence of sensor faults. Specifically, the paper introduces comprehensive frameworks integrating fault-tolerant mechanisms and attack detection with control barrier functions. Section 4.5 is based on Fault Tolerant Neural Control Barrier Functions for Robotic Systems under Sensor Faults and Attacks, which is coauthored with Dr. Luyao Niu, Prof. Andrew Clark and Prof. Radha Poovendran. This paper generalizes the fault-tolerant CBFs to fault-tolerant NCBFs for better approximation of complex environments.

Chapter 5 Resilient Safe Control under LiDAR Perception Attacks: This chapter addresses safety and resilience in the presence of LiDAR Spoofing attacks. Section 5.2 is based on Barrier Certificate based Safe Control for LiDAR-based Systems under Sensor Faults and Attacks, which is coauthored with Mr. Shiyu Cheng, Dr Luyao Niu and Prof. Andrew Clark. In the paper, we focus on 2D-LiDAR-based systems and investigate threats unique to these sensors. A fault detection, isolation and recovery approach is proposed together with control barrier certificate to achieve resilient safe control under LiDAR attacks. Section 5.3 is based on Cooperative Perception for Safe Control of Autonomous Vehicles under LiDAR Spoofing Attacks, which is coauthored with Dr. Zhouchi Li, Mr. Shiyu Cheng, and Prof. Andrew Clark. The paper reveals the characteristics of 3D LiDAR-based perception. It proposes methodologies for fault detection, isolation, and identification (FDII), combined with robust fault-tolerant control (FTC) strategies that ensure the system remains within safe operating regions despite compromised components.

Chapter 2

Neural Control Barrier Functions For Deterministic Systems

Control Barrier Functions (CBFs) are a popular approach for safe control of nonlinear systems. In CBF-based control, the desired safety properties of the system are mapped to nonnegativity of a CBF, and the control input is chosen to ensure that the CBF remains nonnegative for all time. Recently, machine learning methods that represent CBFs as neural networks (neural control barrier functions, or NCBFs) have shown great promise due to the universal representability of neural networks. However, verifying that a learned CBF guarantees safety remains a challenging research problem. This chapter presents novel exact conditions and algorithms for verifying safety of feedforward NCBFs with ReLU activation functions.

The key challenge in doing so is that, due to the piecewise linearity of the ReLU function, the NCBF will be nondifferentiable at certain points, thus invalidating traditional safety verification methods that assume a smooth barrier function. We resolve this issue by leveraging a generalization of Nagumo's theorem for proving invariance of sets with nonsmooth boundaries to derive necessary and sufficient conditions for safety. Based on this condition, we propose an algorithm for safety verification of NCBFs that first decomposes the NCBF into piecewise linear segments, named hyperplanes for differential and hinges for nondifferentiable, respectively.

We propose a VNN-based methods and then solves a nonlinear program to verify safety of each segment as well as the intersections of the linear segments. We mitigate the complexity by only considering the boundary of the safe region and by pruning the segments with Interval Bound Propagation (IBP) and linear relaxation. We evaluate our approach through numerical studies with comparison to state-of-the-art SMT-based methods.

VNN-based verifications exploit the piecewise-linear structure of ReLU neural networks, however, such approaches still rely on enumerating all of the activation regions of the network near the safety boundary, thus incurring high computation cost. To address this issue, we propose a framework for Synthesis with Efficient Exact Verification (SEEV). Our framework consists of two components, namely (i) an NCBF synthesis algorithm that introduces a novel regularizer to reduce the number of activation regions at the safety boundary, and (ii) a verification algorithm that exploits tight over-approximations of the safety conditions to reduce the cost of verifying each piecewise-linear segment. Our simulations show that SEEV significantly improves verification efficiency while maintaining the NCBF quality across various benchmark systems and neural network structures.

Contributions: This chapter resolves the challenge of verifying safety of NCBFs with ReLU activation functions with the following contributions.

- We derive exact safety conditions for ReLU NCBFs by leveraging a generalization of Nagumo's theorem for proving invariance of sets with nonsmooth boundaries.
- We propose an algorithm to verify that an NCBF satisfies our derived safety conditions. Our approach leverages the piece-wise linearity of ReLU neural networks and decomposes the NCBF into hyperplanes and hinges. In order to mitigate the complexity of this stage, we show that it suffices to consider the boundary of the safe region.
- We propose a VNN-based searching algorithm to identify boundary hyperplanes and hinges by over-approximate the input-output relationship of the NCBF with Interval Bound Propagation and linear relaxation. After decomposing the NCBF, we verify safety by solving a set of nonlinear programs
- We mitigate computational bottleneck by (i) developing a training procedure that reduces the number of segments that must be verified and (ii) constructing verification algorithms that efficiently enumerate the linear segments at the safety boundary and exploit easily-checked sufficient conditions to reduce computation time.
- Towards (i), we introduce a regularizer to the loss function that penalizes the dissimilarity of activation patterns along the CBF boundary.
- Towards (ii), we propose a breadth-first search algorithm for efficiently enumerating the boundary segments, as well as tight linear over-approximations of the nonlinear optimization problems for verifying each segment.

- Moreover, we integrate the synthesis and verification components by incorporating safety counterexamples returned by the safety verifier into the training dataset.
- Our simulation results demonstrate significant improvements in verification efficiency and reliability across a range of benchmark systems.

Organization: The remainder of this chapter is organized as follows. Section 2.1 present existing work and how this chapter differentiate from these works. Section 2.2 gives the system model and background on neural networks and notation. Section 2.3 presents the problem formulation and exact conditions for safety. Section 2.4 presents VNN-based algorithm and neural-breadth-first-search approach to identify the hyperplanes and hinges containing the boundary of the NCBF safe region. Section 2.5 present exact and hierarchical approach to effectively verify that an NCBF satisfies the safety conditions. Section 2.6 evaluate our proposed verification method and compare with state-of-the-art verification methods. Section 2.7 presents the framework to synthesize a verifiable ReLU NCBF to ensure the safety of the system. Section 2.8 concludes the chapter.

2.1 Related Work

Energy-based methods have been proposed to guarantee safety by ensuring that a particular energy function remains nonnegative. Barrier certificates for safe control were first proposed in [12]. More recently, CBFs have emerged as promising approaches to safe control, due to their compatibility with a wide variety of control laws [8, 20, 18, 21, 22, 23, 24]. Sum-of-squares (SOS) optimization and other techniques derived from algebraic geometry have been widely used for safety verification of polynomial barrier functions [12, 8, 64, 65, 18]. However, SOS-based approaches for polynomial CBFs cannot be applied directly to NCBF verification, since activation functions used in neural networks are not polynomial and may be non-differentiable.

Neural barrier certificate [66, 40, 67, 68] and NCBFs [23, 16, 22] have been proposed to describe complex safety constraints that cannot be encoded polynomials. Current work, including SMT-based methods [69, 40, 70] and mixed integer programs [71], verify safety by constructing a nominal control policy and proving that it satisfies the NCBF constraints. However, as we show in Section 3.4, the reliance on a particular control policy may lead to false

negatives during safety verification. Another related body of work deals with the problem of verifying neural networks including SMT [72], output reachable set verification [73], polynomial approximations of the barrier function [74], verify input/output relationships [75, 76, 77] and ReLU neural networks focused verification [78, 79]. These methods, however, are not directly applicable to the problem of NCBF verification, which requires joint consideration of the neural network and the underlying nonlinear system dynamics. Methods based on input/output relationship can verify by encode dynamics and control policy with neural networks. However, with approximating error introduced, these methods are not directly applicable for exact verification. Piecewise linear approximations of ReLU neural networks have been used to develop tractable safety verification algorithms using linear [80] and SOS programming [81]. These approaches leads to sound and incomplete verification algorithms and have only be applied for discrete-time systems, whereas the present chapter proposes exact verification algorithms for continuous-time systems.

2.1.1 Verification of NCBFs

Current work, including SMT-based methods [69, 40, 70] and mixed integer programs [71], verifies safety by constructing a nominal control policy and proving that it satisfies the NCBF constraints. However, the reliance on a particular control policy may lead to false negatives during safety verification. Another related body of work deals with the problem of verifying neural networks including SMT [72], output reachable set verification [73], polynomial approximations of the barrier function [74], verify input/output relationships [75, 76, 77] and ReLU neural networks focused verification [78, 79]. These methods, however, are not directly applicable to the problem of NCBF verification, which requires joint consideration of the neural network and the underlying nonlinear system dynamics. Methods based on input/output relationship can verify by encode dynamics and control policy with neural networks. However, with approximating error introduced, these methods are not directly applicable for exact verification. Piecewise linear approximations of ReLU neural networks have been used to develop tractable safety verification algorithms using linear [80] and SOS programming [81]. These approaches leads to sound and incomplete verification algorithms and have only be applied for discrete-time systems, whereas the present chapter proposes exact verification algorithms for continuous-time systems.

A key challenge in NCBF-based control is safety verification, which amounts to ensuring that the constraints on the control can be satisfied throughout the state space under actuation limits. The NCBF safety verification problem effectively combines two problems that are known to be difficult, namely, input-output verification of neural networks (VNN) [82, 83, 84, 85, 86, 87] and reachability verification of nonlinear systems. While sound and complete verifiers such as dReal can be applied to NCBFs, they typically can only handle systems of dimension three or small neural networks [40, 88]. In [39], exact conditions for safety verification of NCBFs with ReLU activation functions were proposed that leverage the piecewise-linearity of ReLU-NNs to reduce verification time compared to dReal for general activation functions. The exact conditions, however, still require checking correctness of the NCBF by solving a nonlinear optimization problem along each piecewise-linear segment. Hence, the NCBF verification problem remains intractable for high-dimensional systems.

2.1.2 Synthesis of Verifiable Safe Control

Neural control barrier functions have been proposed to describe complex safety sets to remain inside and certify safety of a controlled system [66, 40, 67, 68] or synthesize control input based on NCBFs to ensure safety [16, 89, 23, 22]. However, the synthesized NCBF may not ensure safety. Safety verification of NCBFs is required. Sum-of-squares (SOS) optimization [12, 8, 90, 65, 18] has been widely used for polynomial barrier functions, however, they are not applicable due to the non-polynomial and potentially non-differentiable activation functions of NCBFs.

Verification-in-the-loop approaches have been proposed to synthesize neural networks with verifiable guarantees from reachability analysis [42] and neural network verification [91]. Counterexample Guided Inductive Synthesis (CEGIS) has been applied using SMT-based techniques [40, 69, 70, 92, 93]. However, SMT-based methods do not scale well with increasing network size. Other verification-in-the-loop approaches utilize reachability analysis [42] and branch-and-bound neural network verification tools [91]. However, existing works suffer from the difficulty of performing verification and generating counterexamples in a computationally efficient manner. Sampling-based approaches [93, 35] aim to prove safety using Lipschitz conditions, but they rely on dense sampling over the state space, which is computationally prohibitive.

2.2 Problem Formulation

2.2.1 System Model

We consider a continuous-time nonlinear control-affine system with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, control input $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$, and dynamics

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

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are known functions. A control policy is a function $\mu : \mathbb{R}^n \to \mathcal{U}$ that maps a state x to a control input u.

2.2.2 Safety and Control Barrier Functions

Safety of dynamical systems requires $x(t)$ to remain in a given region $\mathcal{C}$, which we denote as the safe region. We assume the safe region is given by $\mathcal{C} = \{x : h(x) \geq 0\} \subseteq \mathcal{X}$, for some function $h : \mathcal{X} \to \mathbb{R}$. Safety is related to the property of positive invariance, which we define as follows.

Definition 2.1. *A set $\mathcal{D} \subseteq \mathbb{R}^n$ is positive invariant under dynamics (2.1) and control policy μ if $x(0) \in \mathcal{D}$ and $u(t) = \mu(x(t))\ \forall t \geq 0$ imply that $x(t) \in \mathcal{D}$ for all $t \geq 0$.*

We define a control policy μ to be *safe* if there is a set $\mathcal{D}$ such that (i) $\mathcal{D} \subseteq \mathcal{C}$ and (ii) $\mathcal{D}$ is positive invariant under dynamics (2.1) and control policy μ.

One approach to designing safe control policies is to choose a function $b : \mathbb{R}^n \to \mathbb{R}$, denoted as a *Control Barrier Function (CBF)*, and let $\mathcal{D} = \{x : b(x) \geq 0\}$. In the case where b is continuously differentiable, the following result can be used to guarantee positive invariance of $\mathcal{D}$.

Theorem 2.1 ([8])**.** *Suppose that b is a CBF, $b(x(0)) \geq 0$, and $u(t)$ satisfies*

$$\frac{\partial b}{\partial x}(f(x) + g(x)u) \geq -\alpha(b(x)). \tag{2.2}$$

for all t, where $\alpha : \mathbb{R} \to \mathbb{R}$ is a strictly increasing function with $\alpha(0) = 0$. Then the set $\mathcal{D} = \{x : b(x) \geq 0\}$ is positive invariant.

Theorem 2.1 implies that, if b is a CBF, then adding (2.2) as a constraint on the control at each time step suffices to guarantee safety.

2.2.3 Problem Formulation

We consider a feedforward neural network (NN) $b : \mathcal{X} \to \mathbb{R}$ with ReLU activation function. Since b is not continuously differentiable due to the piecewise linearity of the ReLU function, the guarantees of Theorem 2.1 cannot be applied directly. Our overall goal will be to derive analogous conditions to (2.2) for ReLU-NCBFs, and then ensure that any control policy μ that satisfies the conditions will be safe with $\mathcal{D} = \{x : b(x) \geq 0\}$.

Problem 2.1. *Given a nonlinear continuous-time system (2.1), a neural network function $b : \mathcal{X} \to \mathbb{R}$, a set of admissible control inputs $\mathcal{U} = \{u : Au \leq c\}$ for given matrix $A \in \mathbb{R}^{p \times m}$ and vector $c \in \mathbb{R}^p$, and a safe set $\mathcal{C} = \{x : h(x) \geq 0\}$, determine whether (i) $\mathcal{D} \subseteq \mathcal{C}$ and (ii) there exists a control policy μ such that $\mathcal{D}$ is positive invariant under dynamics (2.1) and control policy μ.*

2.3 Exact Conditions for Safety

Nagumo's Theorem gives necessary and sufficient conditions for positive invariance of a set. We first define the concept of tangent cone, and then present positive invariance conditions based on the tangent cone. The approach of the proof is to characterize the tangent cone to the set $\mathcal{D} = \{x : b(x) \geq 0\}$.

Definition 2.2. *Let $\mathcal{A}$ be a closed set. The tangent cone to $\mathcal{A}$ at x is defined by*

$$\mathcal{T}_{\mathcal{A}}(x) = \left\{ z : \liminf_{\tau \to 0} \frac{dist(x + \tau z, \mathcal{A})}{\tau} = 0 \right\} \tag{2.3}$$

The following result gives an approach for constructing the tangent cone.

Lemma 2.1 ([94]). *Suppose that the set $\mathcal{A}$ is defined by*

$$\mathcal{A} = \{x : q_k(x) \leq 0, k = 1, \ldots, N\}$$

for some collection of differentiable functions $q_1, \ldots, q_N$. For any x, let $J(x) = \{k : q_k(x) = 0\}$. Then

$$\mathcal{T}_{\mathcal{A}}(x) = \{z : z^T \nabla q_k(x) \leq 0 \ \forall k \in J(x)\}.$$

The following is a fundamental preliminary result for establishing positive invariance.

Theorem 2.2 (Nagumo's Theorem [94], Section 4.2). *A closed set $\mathcal{A}$ is controlled positive invariant if and only if, whenever $x(t) \in \partial\mathcal{A}$, $u(t) \in \mathcal{U}$ satisfies*

$$(f(x(t)) + g(x(t))u(t)) \in \mathcal{T}_{\mathcal{A}}(x(t)) \tag{2.4}$$

When a CBF $b(x)$ is continuously differentiable, ensuring that (2.2) is satisfied is equivalent to verifying that there is no x satisfying $b(x) = 0$, $\frac{\partial b}{\partial x} g(x) = 0$, and $\frac{\partial b}{\partial x} f(x) < 0$ [18].

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

Lemma 2.2 (Farkas' Lemma [95]). *Let A be a real matrix with m rows and n columns, and let $b \in \mathbb{R}^n_u$ be a vector. One of the following conditions holds.*

- *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.*

2.3.1 Safety Violation due to Non-differentiability

When b is represented by a ReLU neural network, however, b will not be differentiable when the input x leads to neurons having zero pre-activation input, i.e., when $\mathbf{T}(x) \neq \emptyset$. Although the set of x with $\mathbf{T}(x) \neq \emptyset$ has measure zero, such points can nonetheless cause safety violations as illustrated by the following example.

Example 2.1. *Let $x(t) \in \mathbb{R}^2$ and suppose the dynamics of x are given by*

$$\begin{array}{rcl} \dot{x}_1(t) &=& x_1 + u \\ \dot{x}_2(t) &=& -x_1 + 5x_2 \end{array} \quad \Rightarrow \quad f(x) = \begin{pmatrix} 1 & 0 \\ -1 & 5 \end{pmatrix} x, \; g(x) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Suppose that $\mathcal{U} = \mathbb{R}^m$, the safe region $\mathcal{C} = \{x : x_1^2 + x_2^2 \leq 9\}$, and the candidate CBF is given by $b(x) = 1 - ||x||_1$. This CBF can be realized by a neural network with a single layer ($L = 1$) with four neurons ($M_1 = 4$), weights $W_{11} = (1\ 0)^T$, $W_{12} = (-1\ 0)^T$, $W_{13} = (0\ 1)^T$, $W_{14} = (0\ -1)^T$, $r_{1j} = 0$ for $j = 1, \ldots, 4$, $\Omega_j = -1$ for $j = 1, \ldots, 4$, and $\psi = 1$. We observe that $\mathcal{D} = \{x : b(x) \geq 0\} \subseteq \mathcal{C}$. Furthermore, whenever $\frac{\partial b}{\partial x}$ exists, we have $\frac{\partial b}{\partial x} \in \{(1\ 1), (1\ -1), (-1\ 1), (-1\ -1)\}$, each of which satisfies $\frac{\partial b}{\partial x} g(x) \neq 0$. On the other hand, the set $\mathcal{D}$ is not positive invariant. If $x_2(0) > \frac{1}{5}$, then $|x_1(0)| \leq 1$ implies that $\dot{x}_2(0) > 0$, and indeed, $x_2(t)$ will continue to increase until $x(t) \notin \mathcal{D}$.

Let b_c denote the NCBF defined in the example, which fails our defined safety conditions. For comparison, we trained an NCBF b_θ and verified it using our proposed approach. We then constructed a nominal controller μ_{nom} as a Linear Quadratic Regulator (LQR) controller that drives the system from initial point $(0, 0.1)$ to the origin. We compared the trajectories arising from the optimization-based controller defined by Eq. (10) using the b_θ and b_c. For the unsafe NCBF b_c, the optimization-based controller is unable to satisfy the safety constraints at the boundary point $(0, 1)$, resulting in a safety violation. On the other hand, while the NCBF b_θ contained multiple non-differentiable points, it is possible to choose u to ensure safety at these points. For example, the point $(-0.19, 2.91)$ is a non-differentiable point on the boundary $b_\theta = 0$. There are four activation sets intersecting at this point, with corresponding values of $\frac{\partial b_c}{\partial x} g(x)$ given by $\{-0.0455, -0.053, -0.025, -0.033\}$. Since any control input u with negative sign and sufficiently large magnitude will satisfy $\frac{\partial b_c}{\partial x}(f(x) + g(x)u) \geq 0$ for all of these values, this non-differentiable point does not compromise safety of the system, and the trajectory of the system constrained by b_θ remains in the safe region for all time.

Fundamentally, this safety violation occurs because at $x = (0\ 1)^T$, we have $b(x) = 0$, $\mathbf{T}(x) = \{(1, 1), (1, 2)\}$ (i.e., the preactivation input to the first and second neurons in the hidden layer is zero), creating a discontinuity in the slope of $b(x)$. For x' in the neighborhood of $(0\ 1)^T$, the value of $\frac{\partial b}{\partial x}(x') g(x')$ will be either 1 or -1. Since there is no single control input that satisfies (2.2) for both values of $\frac{\partial b}{\partial x} g(x')$, it is impossible to ensure safety of the system in the neighborhood of $(0\ 1)^T$.

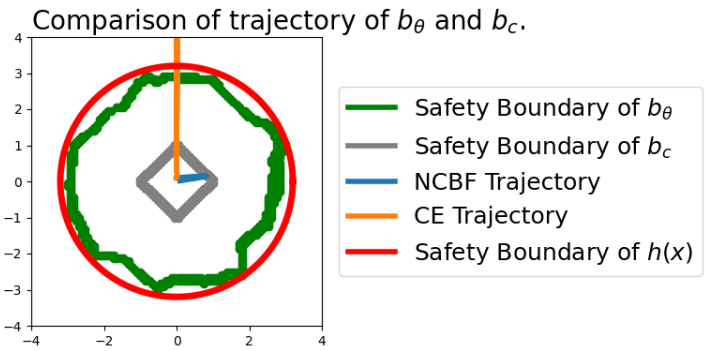


Figure 2.1: Comparison of optimization-based controller using trained NCBF b_θ and unsafe NCBF b_c.

2.3.2 ReLU Neural Control Barrier Function

We give notations to describe a neural network (NN) with L layers and M_i neurons in the i-th layer. We let the input to the NN be denoted x, the output at the j-th neuron of the i-th layer be denoted z_{ij}, and the output of the network be denoted y. We let $\mathbf{z}_i$ denote the vector of neuron outputs at the i-th layer. The outputs are computed as

$$z_{ij} = \begin{cases} \sigma(W_{ij}^T x + r_{ij}), & i = 1 \\ \sigma(W_{ij}^T \mathbf{z}_{i-1} + r_{ij}), & i \in \{2, \dots, L-1\} \end{cases}, \quad y = \Omega^T \mathbf{z}_L + \psi \tag{2.5}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function. The input to the function σ is the *pre-activation input* to the neuron, and is given by $W_{1j}^T x + r_{1j}$ for the j-th neuron at the first layer and $W_{ij}^T \mathbf{z}_{i-1} + r_{ij}$ for the j-th neuron at the i-th hidden layer. W_{ij} has dimensionality $n \times 1$ for $i = 1$ and $M_{i-1} \times 1$ for $i > 1$. W_i is an $n \times M_i$ matrix. In this chapter, we assume that σ is the ReLU function $ReLU(z) = \max\{0, z\}$. The output of the network is given by $y = \Omega^T \mathbf{z}_L + \psi$, where $\Omega \in \mathbb{R}^{M_L}$ and $\psi \in \mathbb{R}$. The j-th neuron at the i-th layer is *activated* by a particular input x if its pre-activation input is nonnegative, *inactivated* if the pre-activation input is nonpositive, and *unstable* if the pre-activation input is zero. A set of neurons $\mathbf{S} = (S_1, \dots, S_L)$, with $S_i \subseteq \{1, \dots, M_i\}$ denoting a subset of neurons at the i-th layer, is activated by x if all of the neurons in $\mathbf{S}$ are activated by x and all the neurons not in $\mathbf{S}$ are inactivated by x. A set of neurons $\mathbf{T} = (T_1, \dots, T_L)$ with $T_i \subseteq \{1, \dots, M_i\}$ is unstable by x if all of the neurons in $\mathbf{T}$ are unstable by x.

For a given set $\mathbf{S}$, if $\mathbf{S}$ is activated by x, then the pre-activation input to each neuron and the overall output of the network are affine in x, with the affine mapping determined by $\mathbf{S}$ as follows. For the first layer, we define

$$\overline{W}_{1j}(\mathbf{S}) = \begin{cases} W_{1j}, & j \in S_1 \\ 0, & \text{else} \end{cases} \quad \overline{r}_{1j}(\mathbf{S}) = \begin{cases} r_{1j}, & j \in S_1 \\ 0, & \text{else} \end{cases} \tag{2.6}$$

so that the output of the j-th neuron at the first layer is $\overline{W}_{1j}(\mathbf{S})^T x + \overline{r}_{1j}(\mathbf{S})$. 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 $\overline{W}_{i1}(\mathbf{S}), \ldots, \overline{W}_{iM_i}(\mathbf{S})$ and

$$\overline{W}_{ij}(\mathbf{S}) = \begin{cases} \overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij}, & j \in S_i \\ 0, & \text{else} \end{cases} \quad \overline{r}_{ij}(\mathbf{S}) = \begin{cases} W_{ij}^T \overline{\mathbf{r}}_{i-1}(\mathbf{S}) + r_{ij}, & j \in S_i, \\ 0, & \text{else} \end{cases} \tag{2.7}$$

where $\overline{\mathbf{r}}_i(\mathbf{S})$ is the vector with elements $\overline{r}_{ij}(\mathbf{S})$, $j = 1, \ldots, M_i$.

We define $\overline{W}(\mathbf{S}) = \overline{\mathbf{W}}_L(\mathbf{S})\Omega$ and $\overline{r}(\mathbf{S}) = \Omega^T \overline{\mathbf{r}}_L(\mathbf{S}) + \psi$. Based on these notations, when an input x activates the set $\mathbf{S}$, $z_{ij} = \overline{W}_{ij}(\mathbf{S})^T x + \overline{r}_{ij}(\mathbf{S})$ and $y = \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S})$.

Lemma 2.3. *Let $\overline{\mathcal{X}}(\mathbf{S})$ denote the set of inputs x that activate a particular set of neurons $\mathbf{S}$, and let $\overline{\mathbf{W}}_0(\mathbf{S})$ be equal to the identity matrix, and $\overline{\mathbf{r}}_0$ to be zero vector. Then*

$$\overline{\mathcal{X}}(\mathbf{S}) = \bigcap_{i=1}^{L} \left(\bigcap_{j \in S_i} \{x : W_{ij}^T (\overline{\mathbf{W}}_{i-1}(\mathbf{S})^T x + \overline{\mathbf{r}}_{i-1}) + r_{ij} \geq 0\} \right.$$
$$\left. \cap \bigcap_{j \notin S_i} \{x : W_{ij}^T (\overline{\mathbf{W}}_{i-1}(\mathbf{S})^T x + \overline{\mathbf{r}}_{i-1}) + r_{ij} \leq 0\} \right). \tag{2.8}$$

Proof. We prove by induction on L. If $L = 1$, then $x \in \overline{\mathcal{X}}(\mathbf{S})$ if the pre-activation input to the $(1, j)$ neuron is nonnegative for all $j \in S_1$ and nonpositive for all $j \notin S_1$. We have that the pre-activation input is equal to $W_{1j}^T x + r_{1j}$, establishing the result for $L = 1$.

Now, inducting on L, we have that $x \in \overline{\mathcal{X}}(S_1, \ldots, S_{L-1})$ if and only if

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

by induction. If $x \in \overline{\mathcal{X}}(S_1, \ldots, S_{L-1})$, then $x \in \overline{\mathcal{X}}(S_L)$ if and only if the pre-activation input to the j-th neuron at layer L is nonnegative for all $j \in S_L$ and nonpositive for $j \notin S_L$. The pre-activation input is equal to $W_{Lj}^T z_{L-1} + r_{Lj}$, which we can expand by induction as

$$\begin{aligned}
W_{Lj}^T z_{L-1} + r_{Lj} &= \sum_{j'=1}^{M_{L-1}} (W_{Lj})_{j'} z_{L-1,j'} + r_{Lj} \\
&= \sum_{j'=1}^{M_{L-1}} (W_{Lj})_{j'} (\overline{\mathbf{W}}_{L-1,j'}(\mathbf{S})^T x + \overline{\mathbf{r}_{L-1,j'}}(\mathbf{S}) + r_{Lj} \\
&= \left(\sum_{j'=1}^{M_{L-1}} (W_{Lj})_{j'} \overline{W}_{L-1,j'}(\mathbf{S}) \right)^T x + \overline{\mathbf{r}}_{Lj}(\mathbf{S}) \\
&= (\overline{\mathbf{W}}_{L-1}(\mathbf{S}) W_{Lj})^T x + \overline{\mathbf{r}}_{Lj}(\mathbf{S})
\end{aligned}$$

completing the proof. □

Note that in (2.8), a particular input x could belong to multiple activation regions $\overline{\mathcal{X}}(\mathbf{S})$. This is because if the j-th neuron at the i-th layer is unstable, then both $(S_1, \ldots, S_i \cup \{j\}, \ldots, S_L)$ and $(S_1, \ldots, S_i \backslash \{j\}, \ldots, S_L)$ can be regarded as activated by x. We let $\mathbf{S}(x) \triangleq \{\mathbf{S} : x \in \overline{\mathcal{X}}(\mathbf{S})\}$ and let $\mathbf{T}(x)$ denote the set of unstable neurons produced by input x. Given a collection of activation sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$, we let

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

The set $\mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$ is equal to the set of neurons that must be unstable in order for an input x to belong to $\overline{\mathcal{X}}(\mathbf{S}_1) \cap \ldots \cap \overline{\mathcal{X}}(\mathbf{S}_r)$.

Finally, we define the terms *hyperplane* and *hinge*. For any $\mathbf{S} \subseteq \{1, \ldots, L\} \times \{1, \ldots, M_i\}$, we define $\overline{\mathcal{X}}(\mathbf{S}) := \{x : \mathbf{S}(x) = \mathbf{S}\}$. The collection of $\overline{\mathcal{X}}(\mathbf{S})$ for all $\mathbf{S}$ is the set of *hyperplanes* associated with the ReLU neural network. A hyperplane that intersects the set $\{x : b_\theta(x) = 0\}$ is a *boundary hyperplane*. The intersection of hyperplanes $\overline{\mathcal{X}}(\mathbf{S}_1), \ldots, \overline{\mathcal{X}}(\mathbf{S}_r)$ is called a *hinge*. A hinge that intersects the set $\{x : b_\theta(x) = 0\}$ is a *boundary hinge*.

2.3.3 Generalized Nagumo's Theory for ReLU NCBF

To address this issue, we can utilize the piece-wise linearity of ReLU neural networks by regarding the ReLU NCBF as a set of linear CBFs and intersections among linear CBFs.

Proposition 2.1. *For any $x \in \partial\mathcal{D}$, we have*

$$\mathcal{T}_{\mathcal{D}}(x) = \bigcup_{\mathbf{S} \in \mathbf{S}(x)} \left[\left(\bigcap_{(i,j) \in \mathbf{T}(x) \cap \mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T z \geq 0\} \right) \cap \left(\bigcap_{(i,j) \in \mathbf{T}(x) \setminus \mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T z \leq 0\} \right) \cap \{z : \overline{W}(\mathbf{S})^T z \geq 0\} \right] \tag{2.9}$$

Proof. Define $\overline{\mathcal{X}}_0(\mathbf{S}) = \overline{\mathcal{X}}(\mathbf{S}) \cap \mathcal{D}$. We will first show that, for all x with $b(x) = 0$,

$$\mathcal{T}_{\mathcal{D}}(x) = \bigcup_{\mathbf{S} \in \mathbf{S}(\mathbf{x})} \mathcal{T}_{\overline{\mathcal{X}}_0(\mathbf{S})}(x). \tag{2.10}$$

We observe that

$$\operatorname{dist}\left(x, \mathcal{D} \setminus \bigcup_{\mathbf{S} \in \mathbf{S}(x)} \overline{\mathcal{X}}_0(\mathbf{S}) \right) > 0,$$

and hence

$$\operatorname{dist}(x + \tau z, \mathcal{D}) = \min_{\mathbf{S} \in \mathbf{S}(x)} \operatorname{dist}(x + \tau z, \overline{\mathcal{X}}_0(\mathbf{S}))$$

for τ sufficiently small.

Suppose that $z \in \mathcal{T}_{\overline{\mathcal{X}}_0(\mathbf{S})}(x)$. Then for any $\tau \geq 0$, $\text{dist}(x+\tau z, \mathcal{D}) \leq \text{dist}(x+\tau z, \overline{\mathcal{X}}_0(\mathbf{S}))$ since $\overline{\mathcal{X}}_0(\mathbf{S}) \subseteq \mathcal{D}$, and hence

$$\liminf_{\tau \to 0} \frac{\text{dist}(x+\tau z, \mathcal{D})}{\tau} \leq \liminf_{\tau \to 0} \frac{\text{dist}(x+\tau z, \overline{\mathcal{X}}_0(\mathbf{S}))}{\tau} = 0.$$

We therefore have $z \in \mathcal{T}_{\mathcal{D}}(x)$.

Now, suppose that $z \in \mathcal{T}_{\mathcal{D}}(\mathbf{x})$ and yet $z \notin \bigcup_{\mathbf{S} \in \mathbf{S}(x)} \mathcal{T}_{\overline{\mathcal{X}}_0(\mathbf{S})}(x)$. Then for all $\mathbf{S} \in \mathbf{S}(x)$, there exists $\epsilon_{\mathbf{S}} > 0$ such that

$$\liminf_{\tau \to 0} \frac{\text{dist}(x+\tau z, \overline{\mathcal{X}}_0(\mathbf{S}))}{\tau} = \epsilon_{\mathbf{S}}.$$

Let $\bar{\epsilon} = \min\{\epsilon_{\mathbf{S}} : \mathbf{S} \in \mathbf{S}(x)\}$. For any $\delta \in (0, \bar{\epsilon})$, there exists $\overline{\tau} > 0$ such that $\tau < \overline{\tau}$ implies

$$\frac{\text{dist}(x+\tau z, \mathcal{D})}{\tau} = \min_{\mathbf{S} \in \mathbf{S}(x)} \frac{\text{dist}(x+\tau z, \overline{\mathcal{X}}_0(\mathbf{S}))}{\tau} > \delta$$

implying that $\liminf_{\tau \to 0} \frac{\text{dist}(x+\tau z, \mathcal{D})}{\tau} > 0$ and hence $z \notin \mathcal{T}_{\mathcal{D}}(x)$. This contradiction implies (2.10).

It now suffices to show that, for each $\mathbf{S} \in \mathbf{S}(x)$,

$$\mathcal{T}_{\overline{\mathcal{X}}_0(\mathbf{S})}(x) = \left(\bigcap_{(i,j) \in \mathbf{T}(x) \cap \mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T z \geq 0\} \right) \cap \left(\bigcap_{(i,j) \in \mathbf{T}(x) \setminus \mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T z \leq 0\} \right) \cap \{z : \overline{W}(\mathbf{S})^T z \geq 0\}.$$

We have that each $\overline{\mathcal{X}}_0(\mathbf{S})$ is given by

$$\begin{aligned} \overline{\mathcal{X}}_0(\mathbf{S}) = &\{x' : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T x' + r_{ij}(\mathbf{S}) \geq 0 \ \forall (i,j) \in \mathbf{S}\} \\ &\cap \{x' : (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T x' + r_{ij}(\mathbf{S}) \leq 0 \ \forall (i,j) \notin \mathbf{S}\} \cap \{x' : \overline{W}(\mathbf{S})^T x' + r(\mathbf{S}) \geq 0\}, \end{aligned}$$

thus matching the conditions of Lemma 2.1 when each g_k function is affine. Furthermore, the set $J(x)$ is equal to the set of functions that are exactly zero at x, which consists of $\{(\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}) : (i,j) \in T(\mathbf{x})\}$ together with $\overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S})$. This observation combined with Lemma 2.1 gives the desired result. □

Lemma 2.4 is a consequence of Proposition 2.1. The following lemma addresses this issue by giving exact and general conditions for a NCBF with ReLU activation function to satisfy positive invariance. We let $\partial\mathcal{D}$ denote the boundary of the set $\mathcal{D}$.

Lemma 2.4. *The set $\mathcal{D}$ is positive invariant if and only if, for all $x \in \partial\mathcal{D}$, there exist $\mathbf{S} \in \mathbf{S}(x)$ and $u \in \mathcal{U}$ satisfying*

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S})W_{ij})^T(f(x)+g(x)u) \geq 0 \ \forall (i,j) \in \mathbf{T}(x) \cap \mathbf{S} \tag{2.11}$$

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S})W_{ij})^T(f(x)+g(x)u) \leq 0 \ \forall (i,j) \in \mathbf{T}(x) \setminus \mathbf{S} \tag{2.12}$$

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S})W_{ij})^T(f(x)+g(x)u) \geq 0 \tag{2.13}$$

Proof. By Theorem 2.2, the set $\mathcal{D}$ is positive invariant if and only if for every $x \in \partial\mathcal{D}$, there exists u such that $(f(x)+g(x)u) \in \mathcal{T}_{\mathcal{D}}(x)$. By Proposition 2.1, this condition holds iff there exists $\mathbf{S} \in \mathbf{S}(x)$ such that

$$(f(x)+g(x)u) \in \left(\left(\bigcap_{(i,j)\in\mathbf{T}(x)\cap\mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S})W_{ij})^T z \geq 0\} \right) \cap \left(\bigcap_{(i,j)\in\mathbf{T}(x)\setminus\mathbf{S}} \{z : (\overline{\mathbf{W}}_{i-1}(\mathbf{S})W_{ij})^T z \leq 0\} \right) \cap \{z : \overline{W}(\mathbf{S})^T z \geq 0\} \right)$$

The above condition is equivalent to the conditions of the lemma, completing the proof. □

Then the verification problem can be reformulated into verifying each activation set and intersections. We now present the sufficient and necessary condition of a feasible NCBF in the following Lemma.

Intuitively, conditions (2.11)–(2.13) can be interpreted as follows. Condition (2.13) is similar to (2.2) when the gradient is given by $\overline{W}(\mathbf{S})$, i.e., when x is in the interior of the activation region defined by $\mathbf{S}$. Eq. (2.11)–(2.12) can be interpreted as choosing the control input to ensure that x remains in the activation region of $\mathbf{S}$ (i.e., $\overline{\mathcal{X}}(\mathbf{S})$).

Lemma 2.4 can be used to construct NCBF-based safe control policies, analogous to control policies based on continuously differentiable CBFs. Formally, given a nominal control policy

μ_{nom}, at each time t, we can choose $u(t)$ by solving the optimization problem

$$\begin{array}{ll} \min_{\mathbf{S}\in\mathbf{S}(x),u} & ||u-\mu_{nom}(x(t))||_2^2 \\ \text{s.t.} & \overline{W}(\mathbf{S})^T(f(x)+g(x)\mu(x)) \geq -\alpha(b(x)) \\ & u\in\mathcal{U}, (2.11), (2.12) \end{array} \tag{2.14}$$

where α satisfies the same conditions as in Theorem 2.1. This problem can be solved by decomposing (2.14) into $|\mathbf{S}(x)|$ quadratic programs (one for each $\mathbf{S}$) and then selecting the value of u that minimizes the objective function across all of the QPs. In particular, if $\mathbf{S}(x)$ is a singleton, then the constraints on (2.14) reduce to Eq. (2.2).

We then give an equivalent condition for $b(x)$ to be an NCBF. As a preliminary, we say that a collection of activation sets $\mathbf{S}_1,\dots,\mathbf{S}_r$ is *complete* if for any $\mathbf{S}'\notin\{\mathbf{S}_1,\dots,\mathbf{S}_r\}$, we have $\overline{\mathcal{X}}(\mathbf{S}_1)\cap\cdots\cap\overline{\mathcal{X}}(\mathbf{S}_r)\cap\overline{\mathcal{X}}(\mathbf{S}')=\emptyset$.

Proposition 2.2. *The function b is a valid CBF iff the following two properties hold:*

(i) For all activation sets $\mathbf{S}_1,\dots,\mathbf{S}_r$ with $\{\mathbf{S}_1,\dots,\mathbf{S}_r\}$ complete and any x satisfying $b(x)=0$ and

$$x\in\left(\bigcap_{l=1}^{r}\overline{\mathcal{X}}(\mathbf{S}_l)\right), \tag{2.15}$$

there exist $l\in\{1,\dots,r\}$ and $u\in\mathcal{U}$ such that

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T(f(x)+g(x)u)\geq 0 \quad \forall(i,j)\in\mathbf{T}(\mathbf{S}_1,\dots,\mathbf{S}_r)\cap\mathbf{S}_l \tag{2.16}$$

$$(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T(f(x)+g(x)u)\leq 0 \quad \forall(i,j)\in\mathbf{T}(\mathbf{S}_1,\dots,\mathbf{S}_r)\setminus\mathbf{S}_l \tag{2.17}$$

$$\overline{W}(\mathbf{S}_l)^T(f(x)+g(x)u)\geq 0 \tag{2.18}$$

(ii) For all activation sets $\mathbf{S}$, we have

$$(\overline{\mathcal{X}}(\mathbf{S})\cap\mathcal{D})\setminus\mathcal{C}=\emptyset \tag{2.19}$$

The proof can be found in the supplementary material. We refer to any x that fails to meet at least one of conditions (i) and (ii) of Proposition 2.2 as a safety counterexample.

2.4 Decomposition of ReLU NCBF

The ReLU NCBF can be decomposed into hyperplanes and hinges. Safety counterexamples may be present in hyperplanes or hinges containing the zero-level set of the NCBF. In this section, we present two search algorithms i) VNN-based approach utilize LiRPA conducting coarser-to-finer search by discretizing the state space and ii) Neural Breadth-First-Search searching for activated sets using linear programs in a Breadth-First-Search manner.

2.4.1 VNN-based Search Algorithm

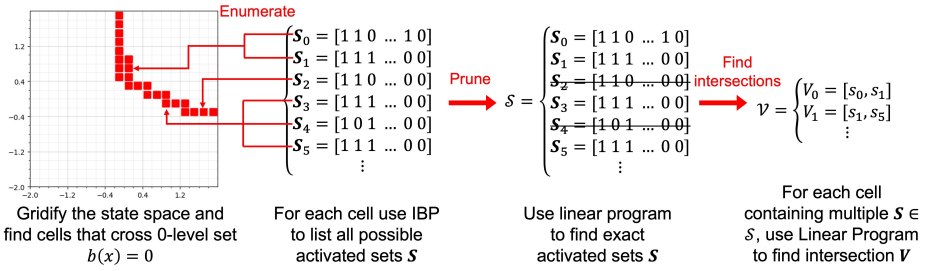


Figure 2.2: Illustration of proposed coarser-to-finer searching method. Hyper-cubes that intersect the safety boundaries are marked in red. When all possible activation sets are listed, we can identify exact activation set and intersections.

The preceding proposition motivates our overall approach to verifying NCBFs, consisting of the following steps. **(Step 1)** We conduct coarser-to-finer search by discretizing the state space into hyper-cubes and use linear relaxation based perturbation analysis (LiRPA) to identify grid squares that intersect the boundary $\{x : b(x) = 0\}$. **(Step 2)** We enumerate all possible activation sets within each hyper-cube using Interval Bound Propagation (IBP). We then identify the activation sets and intersections that satisfy $b(x) = 0$ using linear programming. **(Step 3)** For each activation set and intersection of activation sets, we verify the conditions of Proposition 2.2. In what follows, we describe each step in detail.

We next present the approach to enumerate activation sets of a given ReLU NCBF that intersect the safety boundary, $b(x) = 0$. Our approach first conducts a coarse-grained search that over-approximates the collection of activation sets that intersect with the safety boundary. We then prune this collection to remove activation sets that cannot be realized. Specifically, we first identify regions that contain $b(x) = 0$, then enumerate all possible activation sets within the region and finally identify the sets that intersect the safety boundaries.

We first discretize the given state space $\mathcal{X}$ into hyper-cubes. We compute lower and upper bounds of $b(x)$ on each hyper-cube, denoted b_l and b_u respectively, with linear relaxation based perturbation analysis (LiRPA), i.e., auto LiRPA [83]. For each hyper-cube B, we determine B contains $b(x) = 0$ (red squares in Fig 2.2) if criteria $sgn(b_l) \cdot sgn(b_u) \leq 0$ holds, where $sgn(*)$ is the sign function. For each B contains $b(x) = 0$, we utilize IBP to over-approximate unstable neurons.

Interval bound propagation aims to compute an interval of possible output values by propagating a range of inputs layer-by-layer, and is integrated into our approach as follows. We first use partition the state space into cells and, for each cell, use LiRPA to derive upper and lower bounds on the value of b(x) when x takes values in that cell. When the interval of possible b(x) values in a cell contains zero, we conclude that that cell may intersect the boundary b(x) = 0. For each neuron, we use IBP to compute the pre-activation input interval for values of x within the cell. When the pre-activation input has a positive upper bound and negative lower bound, we identify the neuron as unstable, i.e., it may be either positive or negative for values of x within the cell. Using this approach, we enumerate a collection of activation sets $\mathcal{S}$. We then identify the activation sets $\mathbf{S} \in \tilde{\mathcal{S}}$ such that $b(x) = 0$ for some $x \in \overline{\mathcal{X}}(\mathbf{S})$ by searching for an x that satisfies the linear constraints in (16). This approach uses LiRPA and IBP to identify the activation regions that intersect the boundary $\{x : b(x) = 0\}$ without enumerating and checking all possible activation sets, which would have exponential runtime in the number of neurons in the network.

Let $\tilde{\mathcal{S}}$ denote the over-approximated collection of activation sets computed by IBP. Note that $\boldsymbol{S} \in \tilde{\mathcal{S}}$ may not intersect with $b(x) = 0$. As shown in Fig. 2.2, we let $\mathcal{S}$ denote the collection of activation sets $\mathbf{S} \in \tilde{\mathcal{S}}$ that satisfy

$$\begin{aligned} \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) &= 0 \\ (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}) &\geq 0 \ \forall (i,j) \in \mathbf{S} \\ (\overline{\mathbf{W}}_{i-1}(\mathbf{S}) W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}) &\leq 0 \ \forall (i,j) \notin \mathbf{S} \end{aligned} \tag{2.20}$$

for some $x \in B$. The activation set boundaries indicate unstable neurons, which need to be verified separately. Hence, we search for intersections as follows, where $2^{\mathcal{S}}$ denotes the set of subsets of $\mathcal{S}$.

$$\mathcal{V} = \left\{ Z \in 2^{\mathcal{S}} : \left(\bigcap_{\mathbf{S} \in Z} \overline{\mathcal{X}}(\mathbf{S}) \right) \cap \{x : b(x) = 0\} \neq \emptyset \right\}$$

2.4.2 Neural Breadth-First-Search

The boundary enumeration identifies the boundary hyperplanes $\mathbf{S}_0 \in \mathcal{B}$. It initially identifies the initial boundary activation set $\mathbf{S}_0 \in \mathcal{B}$, enumerates all $\mathbf{S} \in \mathcal{B}$ by NBFS starting from $\mathbf{S}_0$, and finally enumerates all hinges consisting of the intersections of hyperplanes. The NBFS approach avoids over-approximation of the set of boundary hyperplanes that may be introduced by, e.g., interval propagation methods, and hence is particularly suited to deep neural networks.

In what follows, we assume that the unsafe region $\mathcal{X} \setminus \mathcal{C}$ and initial safe set $\mathcal{I}$ are connected, and use $\partial\mathcal{D}$ to refer to the connected component of the boundary of $\mathcal{D}$ that separates $\mathcal{I}$ and $\mathcal{X} \setminus \mathcal{C}$. This assumption is without loss of generality since we can always repeat the following procedure for each connected component of $\mathcal{I}$ and $\mathcal{X} \setminus \mathcal{C}$.

Initial Activation Set Identification:

First, we identify the initial boundary activation set $\mathbf{S}_0$. Given $\hat{x}_U \in \mathcal{X} \setminus \mathcal{C}$ and $\hat{x}_\mathcal{I} \in \mathcal{I}$, define a line segment

$$\tilde{L} = \left\{ x \in \mathbb{R}^n \mid x = (1-\lambda)\hat{x}_{\mathcal{X}\setminus\mathcal{C}} + \lambda\hat{x}_\mathcal{I},\ \lambda \in [0,1] \right\} \tag{2.21}$$

The following lemma shows the initial boundary activation set $\mathbf{S}_0$ can always be produced as $\mathbf{S}_0 = \mathbf{S}(\tilde{x})$ for some $\tilde{x} \in \tilde{L}$.

Lemma 2.5. *Given two sample points $\hat{x}_U$, such that $b(\hat{x}_U) < 0$, and $\hat{x}_\mathcal{I}$, such that $b(\hat{x}_U) > 0$, let $\tilde{L}$ denote the line segment connecting these two points. Then, there exists a point $\tilde{x} \in \tilde{L}$ with $b_\theta(\tilde{x}) = 0$.*

Lemma 1 follows from the intermediate value theorem and continuity of $b_\theta(x)$. In order to search for $\mathbf{S}_0$, we choose a sequence of $N_{\tilde{L}}$ points $x_1^0, \ldots, x_{N_{\tilde{L}}}^0 \in \tilde{L}$. For each x_i^0, we check to see if $\overline{\mathcal{X}}(\mathbf{S}(x_i^0)) \cap \partial\mathcal{D} \neq \emptyset$ by solving boundary linear program BoundaryLP($\mathbf{S}(x_i^0)$) (2.22) as follows. Given a state x_i^0, we define the activation set is $\mathbf{S} = \tau_S(x_i^0)$. To determine if the activation set $\mathbf{S} \in \mathcal{B}$, we solve a linear program referred to as the boundary linear program. The program checks the existence of a state $x \in \overline{\mathcal{X}}(\mathbf{S}(x_i^0))$ that satisfies $\overline{W}(\mathbf{S}(x_i^0))^T x + \overline{r}(\mathbf{S}(x_i^0)) = 0$. The

boundary linear program (BoundaryLP($\mathbf{S}(x_i^0)$)) is defined as follows.

$$\text{BoundaryLP}(\mathbf{S}(x_i^0)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S}(x_i^0))^T x + \overline{r}(\mathbf{S}(x_i^0)) = 0 \\ & x \in \overline{\mathcal{X}}(\mathbf{S}(x_i^0)) \end{cases} \tag{2.22}$$

SEEV identifies the initial activation set by conducting a BoundaryLP-based binary search as shown in Algorithm. 1. The algorithm presents a procedure to identify a hyperplane characterized by an initial activation set $\mathbf{S}_0$ that may contain the boundary $\partial\mathcal{D}$. It iterates over pairs of sample states from the unsafe training set $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ and the safe training set $\mathcal{T}_{\mathcal{I}}$. For each pair $(\hat{x}_{\mathcal{X}\setminus\mathcal{C}}, \hat{x}_{\mathcal{I}})$, the algorithm initializes the left and right points of a search interval. It then performs a binary search by repeatedly computing the midpoint x_{mid} and checking the feasibility of the boundary linear program BoundaryLP(x_{mid}). The algorithm terminates when BoundaryLP(x_{mid}) is feasible, returning the activation set $\mathbf{S}(x_{\text{mid}})$ as $\mathbf{S}_0$.

Algorithm 1 Binary Search for $\mathbf{S}_0$

Input: $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$
Output: initial activation set $\mathbf{S}_0$
procedure EnumInit($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$)
    for $\hat{x}_{\mathcal{X}\setminus\mathcal{C}} \in \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ and $\hat{x}_{\mathcal{I}} \in \mathcal{T}_{\mathcal{I}}$ **do** ▷ Loop over all pairs of samples
        $x_{left}, x_{right} \leftarrow \hat{x}_{\mathcal{X}\setminus\mathcal{C}}, \hat{x}_{\mathcal{I}}$
        $x_{mid} = 0.5(x_{left} + x_{right})$
        while $x_{left} < x_{right}$ **do**
            if BoundaryLP(x_{mid}) **then** ▷ Check if the BoundaryLP is feasible
                Return $\mathbf{S}(x_{mid})$ ▷ Return $\mathbf{S}(x_{mid})$ as $\mathbf{S}_0$
            end if
            if $\overline{W}(\mathbf{S}(x'))x_{mid} + \overline{r}(\mathbf{S}(x')) \geq 0$ **then** ▷ Check if $b(x_{mid}) > 0$
                $x_{right} \leftarrow x_{mid}$ ▷ Update the right point
            else
                $x_{left} \leftarrow x_{mid}$ ▷ Update the left point
            end if
        end while
    end for
end procedure

Neural Breadth-First Search for Decomposition

We next describe Neural Breadth-First Search (NBFS) for enumerating all activation sets along the zero-level set, given an initial set $\mathbf{S}_0$. NBFS enumerates a collection of boundary hyperplanes denoted as $\mathcal{S}$ by repeating the following two steps.

Step 1: Given a set $\mathbf{S}$, NBFS identifies a collection of neighbor activation sets denoted as $\tilde{\mathcal{B}}$ as follows. For each (i,j) with $i = 1,\dots,L$ and $j = 1,\dots,M_i$, construct $\mathbf{S}'$ as $\mathbf{S}' = \begin{cases} \mathbf{S} \setminus (i,j), & (i,j) \in \mathbf{S}' \\ \mathbf{S} \cup (i,j), & (i,j) \notin \mathbf{S}' \end{cases}$. We check whether $\mathbf{S}' \in \mathcal{B}$ by solving the linear program $\mathrm{USLP}(\mathbf{S},(i,j))$ (2.23).

NBFS conducts its search in a breadth-first manner. To determine if a neighboring activation set, resulting from a flip in its (i,j) neuron, may contain the boundary, NBFS solves a linear program, referred to as the unstable neuron linear program of $\mathrm{USLP}(\mathbf{S},(i,j))$. This linear program checks the existence of a state $x \in \overline{\mathcal{X}}(\mathbf{S}) \cap \{x : W_{ij}x + r_{ij} = 0\}$ that satisfies $\overline{W}(\mathbf{S})x + \overline{r}(\mathbf{S}) = 0$. The unstable neuron linear program is defined as follows.

$$\mathrm{USLP}(\mathbf{S},(i,j)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0 \\ & W_{ij}(\mathbf{S})^T x + r_{ij}(\mathbf{S}) = 0 \\ & x \in \overline{\mathcal{X}}(\mathbf{S}) \end{cases} \tag{2.23}$$

If there exists such a state x, then $\mathbf{S}'$ is added to $\tilde{\mathcal{B}}$. To further improve efficiency, we employ the simplex algorithm to calculate the hypercube that overapproximates the boundary hyperplane, denoted as $\mathcal{H}(\mathbf{S}) \supseteq \overline{\mathcal{X}}(\mathbf{S})$, and relax the last constraint of (2.22) to $x \in \mathcal{H}(\mathbf{S})$.

Step 2: For each $\mathbf{S}' \in \tilde{\mathcal{B}}$, NBFS determines if the activation region $\overline{\mathcal{X}}(\mathbf{S}')$ is on the boundary (i.e., satisfies $\overline{\mathcal{X}}(\mathbf{S}') \cap \partial\mathcal{D} \neq \emptyset$) by checking the feasibility of $BoundaryLP(\mathbf{S}')$. If there exists such a state x, then $\mathbf{S}'$ is added to $\mathcal{S}$.

This process continues in a breadth-first search manner until all such activation sets on the boundary have been enumerated. When this phase terminates, $\mathcal{S} = \mathcal{B}$. SEEV utilizes NBFS for enumerating all activation sets along the zero-level set in a breadth-first search manner,

starting from an initial set $\mathbf{S}_0$. The algorithm initializes a queue $\mathcal{Q}$ and a set $\mathcal{S}$ with $\mathbf{S}_0$. While $\mathcal{Q}$ is not empty, it dequeues an activation set $\mathbf{S}$ and checks if the set $\mathbf{S} \in \mathcal{B}$ by solving the boundary linear program BoundaryLP($\mathbf{S}$). If so, $\mathbf{S}$ is identified and added to $\mathcal{S}$. The algorithm then explores its neighboring activation sets by flipping each neuron activation (i, j) in $\mathbf{S}$. For each flip, it solves the unstable neuron linear program USLP($\mathbf{S}, (i, j)$). If USLP is feasible, the new activation set $\mathbf{S}'$ obtained by flipping (i, j) is added to $\mathcal{Q}$ for further search on its neighbors. This process continues until all relevant activation sets are explored, resulting in a set $\mathcal{S}$ that contains activation sets potentially on the boundary.

Algorithm 2 Enumerate Activated Sets

Input: $\mathbf{S}_0$
Output: Set of activation sets $\mathcal{S}$
procedure NBFS($\mathbf{S}_0$)
    Initialize queue $\mathcal{Q}$ with initial activation set $\mathbf{S}_0$
    Initialize sets $\mathcal{S}$ with $\mathbf{S}_0$
    while queue $\mathcal{Q}$ is not empty **do**
        Dequeue $\mathbf{S}$ from $\mathcal{Q}$ ▷ Dequeue the first activation set from the queue
        if BoundaryLP($\mathbf{S}$) is feasible **then** ▷ Check if $\mathbf{S}$ is on a boundary activation set
            if $\mathbf{S} \notin \mathcal{S}$ **then** ▷ If $\mathbf{S}$ is not already in $\mathcal{S}$
                Add $\mathbf{S}$ to $\mathcal{S}$
            end if
            for $(i, j) \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}$ **do** ▷ For activation (i, j) of each neurons
                if USLP($\mathbf{S}, (i, j)$) **then** ▷ Check if the neighbor may contain zero-level set
                    Create $\mathbf{S}'$ by flipping activation (i, j)
                    Add $\mathbf{S}'$ to $\mathcal{Q}$ ▷ Add adjacent activation set $\mathbf{S}'$ to the queue
                end if
            end for
        end if
    end while
    Return $\mathcal{S}$
end procedure

Hinge Enumeration:

The verifier of EEV enumerates a collection $\mathcal{V}$ of boundary hinges, where each boundary hinge $\mathbf{V} \in \mathcal{V}$ is a subset $\mathbf{S}_1, \ldots, \mathbf{S}_r$ of $\mathcal{B}$ with $\overline{\mathcal{X}}(\mathbf{S}_1) \cap \overline{\mathcal{X}}(\mathbf{S}_r) \cap \partial\mathcal{D} \neq \emptyset$.

Given $\mathcal{S}_i \subseteq \mathcal{B}$ and $\mathbf{S}$, hinge enumeration filter the set of neighbor activation sets of $\mathbf{S}$ defined as $\mathcal{N}_{\mathbf{S}}(\mathcal{S}_i) := \{\mathbf{S}' : \mathbf{S}'\Delta\mathbf{S} = 1, \mathbf{S}' \in \mathcal{S}_i\}$. Then, hinge enumeration identifies hinges $\mathbf{V}$ by solving linear program $\text{HingeLP}(\mathcal{N}_{\mathbf{S}}^{(d)}(\mathcal{S}_i))$ (2.24) as follows.

$$\text{HingeLP}(\mathcal{N}_{\mathbf{S}}^{(d)}(\mathcal{S}_i)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, \quad \forall \mathbf{S} \in \mathcal{N}_{\mathbf{S}}(\mathcal{S}_i) \\ & x \in \mathcal{X}(\mathbf{S}), \quad \forall \mathbf{S} \in \mathcal{N}_{\mathbf{S}}(\mathcal{S}_i) \end{cases} \tag{2.24}$$

If $\exists x$, hinge enumeration includes the hinge into the set $\mathcal{V} \cup \mathbf{V}$. The efficiency can be further improved by leveraging the sufficient condition verification proposed in Section 2.5.3. The following result describes the completeness guarantees of $\mathcal{S}$ and $\mathcal{V}$ enumerated in Line 5 and 8 of Algorithm 4.

Proposition 2.3. *Let $\mathcal{S}$ and $\mathcal{V}$ denote the output of Algorithm 4. Then the boundary $\partial\mathcal{D}$ satisfies $\partial\mathcal{D} \subseteq \bigcup_{\mathbf{S}\in\mathcal{S}} \overline{\mathcal{X}}(\mathbf{S})$. Furthermore, if $\mathcal{S}$ is complete and $\left(\bigcap_{i=1}^r \overline{\mathcal{X}}(\mathbf{S}_i)\right) \cap \{x : b(x) = 0\} \neq \emptyset$, then $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\} \in \mathcal{V}$.*

Proof. Suppose that $x' \in \partial\mathcal{D} \setminus \left(\bigcup_{\mathbf{S}\in\mathcal{S}} \overline{\mathcal{X}}(\mathbf{S})\right)$, and let x denote the state on $\partial\mathcal{D}$ found by Line 5 of Algorithm 4. Since $\partial\mathcal{D}$ is connected, there exists a path γ with $\gamma(0) = x$ and $\gamma(1) = x'$ contained in $\partial\mathcal{D}$. Let $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_K$ denote a sequence of activation sets with $\mathbf{S}_0 \in \mathbf{S}(x)$, $\mathbf{S}_K \in \mathbf{S}(x')$, and $\mathbf{S}_0$ equal to the set computed at Line 5 of Algorithm 4. Then there exists $i \geq 2$ such that $\mathbf{S}_{i-1} \in \mathcal{S}$ and $\mathbf{S}_i \notin \mathcal{S}$, and there exists t' such that $\gamma(t') \in \overline{\mathcal{X}}(\mathbf{S}_{i-1}) \cap \overline{\mathcal{X}}(\mathbf{S}_i) \cap \{x : b(x) = 0\}$. We then have that $\mathbf{T}(\mathbf{S}_{i-1}, \mathbf{S}_i)$ is a subset of the set $\mathbf{T}$, and hence $\mathbf{S}_i$ will be identified and added to $\mathcal{S}$ at Line 5 of Algorithm 4.

Now, suppose that $\mathbf{S}_1, \ldots, \mathbf{S}_r \in \mathcal{S}$ is complete and $\left(\bigcap_{i=1}^r \overline{\mathcal{X}}_0(\mathbf{S}_i)\right) \cap \{x : b(x) = 0\} \neq \emptyset$. Let $\mathbf{T} = \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$. Then $\mathbf{T}$ is a subset of the sets $\mathbf{S}_1, \ldots, \mathbf{S}_r \in \mathcal{S}$. Since the intersection of the $\overline{\mathcal{X}}(\mathbf{S})$ sets with $\{x : b(x) = 0\}$ is nonempty, $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ is added to $\mathcal{V}$. □

SEEV enumerates all hinges $\mathbf{V} \in \mathcal{V}$ with Algorithm 3. Algorithm 3 outlines a method to enumerate all feasible hinge hyperplanes formed by combinations of activation sets up to size n. The algorithm takes as input a set of activation sets $\mathcal{S}$ and a maximum combination size n. It initializes an empty list to store feasible hinges. For each combination size k from 2 to n, the algorithm iterates over all activation sets in $\mathcal{S}$. For each activation set $\mathbf{S}$, it generates

Algorithm 3 Enumerate Hinges

Input: $\mathcal{S}$, n
Output: $\mathcal{V}$
procedure HINGEENUM($\mathcal{S}, n$)
    Initialize hinge_list as an empty list
    for k from 2 to n **do** ▷ Outer loop to iterate over combination sizes
        for $\mathbf{S} \in \mathcal{S}$ **do** ▷ Iterate over all activation set
            if $k = 2$ **then**
                $\mathcal{V}_c \leftarrow \mathcal{N}_{\mathbf{S}}(\mathcal{S})$
            else
                $\mathcal{V}_c \leftarrow \mathcal{V}^{(k-1)}$
            end if
            for each combination $\mathcal{S}_c \in \mathcal{V}_c$ and $\mathbf{S} \notin \mathcal{S}_c$ **do** ▷ Iterate over combinations
                if $\mathbf{S}$ and $\mathcal{S}_c$ are not adjacent **then**
                    Continue ▷ skip non-adjacent combinations
                end if
                if HingeLP($\mathcal{S}_c \cup \{\mathbf{S}\}$) **then**
                    Add $\mathbf{V} \leftarrow \mathcal{S}_c \cup \{\mathbf{S}\}$ to $\mathcal{V}^{(k)}$ ▷ Add to set of corresponding k
                end if
            end for
        end for
        Add $\mathcal{V}^{(k)}$ to $\mathcal{V}$
    end for
    return hinge_list
end procedure

candidate combinations $\mathcal{V}_c$ based on adjacency—either the set of adjacent activation sets when $k = 2$, or the feasible combinations from the previous iteration when $k > 2$. It then checks each candidate combination $\mathcal{S}_c$ by verifying adjacency and solving the hinge linear program HingeLP($\mathcal{S}_c \cup \{\mathbf{S}\}$). If the HingeLP is feasible, the combination is added to the list of feasible hinges $\mathcal{V}$. This process continues until all combinations up to size n have been examined, resulting in a comprehensive list of feasible hinge hyperplanes that are essential for understanding the intersections of activation regions.

2.5 Verification

In this section, we demonstrate the efficient exact verification of the given NCBF $b_\theta(x)$ to ensure the positive invariance of the set $\mathcal{D}$ under the NCBF-based control policy.

2.5.1 Verification of Hyperplanes

With the activation sets enumerated, the following result gives an equivalent condition for verifying that there is no safety counterexample in the interior of $\overline{\mathcal{X}}(\mathbf{S})$ for a given activation set $\mathbf{S}$.

Lemma 2.6. *There is no safety counterexample in the set* $\overline{\mathcal{X}}(\mathbf{S}) \setminus \bigcup_{\mathbf{S}' \neq \mathbf{S}} \overline{\mathcal{X}}(\mathbf{S})$ *if and only if:*

1. *There do not exist* $x \in \mathcal{X}$ *and* $y \in \mathbb{R}^{p+1}$ *satisfying (a)* $x \in int(\overline{\mathcal{X}}(\mathbf{S}))$*, (b)* $b(x) = 0$*, (c)* $y \geq 0$*, (d)* $y^T \begin{pmatrix} -\overline{W}(\mathbf{S})^T g(x) \\ A \end{pmatrix} = 0$*, and (e)* $y^T \begin{pmatrix} \overline{W}(\mathbf{S})^T f(x) \\ c \end{pmatrix} < 0$*.*
2. *There does not exist* $x \in \mathcal{X}$ *with* $b(x) = 0$*,* $x \in int(\overline{\mathcal{X}}(\mathbf{S}))$*, and* $h(x) < 0$*.*

Proof. Suppose that condition 1 holds. Then for any $x \in \partial\mathcal{D}$ with $\mathbf{S}(x) = \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$, there exists $l \in \{1, \ldots, r\}$ such that $x \in \overline{\mathcal{X}}(\mathbf{S}_l)$ and $u \in \mathcal{U}$ satisfy (2.11) and (2.12). For this choice of u, we have $(f(x) + g(x)u) \in \mathcal{T}_{\mathcal{D}}(x)$ by Proposition 2.1. Hence $\mathcal{D}$ is positive invariant under any control policy consistent with b by Theorem 2.2.

If Condition 2 holds, then there is no x with $b(x) = 0$ and $x \in \text{int}(\overline{\mathcal{X}}(\mathbf{S}))$ such that $x \notin \mathcal{C}$. Hence, there are no counterexamples to condition (ii) of Proposition 2.2. □

Conditions 1 and 2 of Lemma 2.6 can be verified by solving nonlinear programs (see supplementary material for the formulations of these nonlinear programs). Such nonlinear programs are solved for each activation set $\mathbf{S}$ in the collection $\mathcal{S}$ computed during the enumeration phase.

The following corollary describes the special case where there are no constraints on the control, i.e., $\mathcal{U} = \mathbb{R}^m$.

Corollary 2.1. *If $\mathcal{U} = \mathbb{R}^m$, then there is no safety counterexample in the set $\overline{\mathcal{X}}(\mathbf{S}) \setminus \bigcup_{\mathbf{S}' \neq \mathbf{S}} \overline{\mathcal{X}}(\mathbf{S}')$ if and only if (1) there does not exist $x \in \mathcal{X}$ satisfying $x \in int(\overline{\mathcal{X}}(\mathbf{S}))$, $b(x) = 0$, $\overline{W}(\mathbf{S})^T g(x) = 0$, and $\overline{W}(\mathbf{S})^T f(x) < 0$ and (2) there does not exist $x \in \mathcal{X}$ with $b(x) = 0$, $x \in int(\overline{\mathcal{X}}(\mathbf{S}))$, and $h(x) < 0$.*

Under the conditions of Corollary 2.1, the complexity of the verification problem can be reduced. If $g(x)$ is a constant matrix G, then safety is automatically guaranteed if $\overline{W}(\mathbf{S})^T G \neq 0$. If $f(x)$ is linear in x as well, then verification can be performed via a linear program.

2.5.2 Verification of Hinges

With the activation set boundaries enumerated, we now describe an approach for verifying safety of intersections between activation sets.

Lemma 2.7. *Suppose that the sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$ satisfy condition (ii) of Lemma 2.6. There is no safety counterexample in the set*

$$\overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r) \setminus \bigcup_{S' \notin \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}} \overline{\mathcal{X}}(\mathbf{S}')$$

if and only if there do not exist $x \in \mathcal{X}$ and $y_1, \ldots, y_r \in \mathbb{R}^{T+p+1}$, where $T = |\mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)|$, satisfying (a) $x \in int(\overline{\mathcal{X}}(\mathbf{S}_1)) \cap \cdots \cap int(\overline{\mathcal{X}}(\mathbf{S}_r))$, (b) $b(x) = 0$, (c) $y_l \geq 0$ $\forall l$, (d) $\forall l = 1, \ldots, r$, $y_l^T \Theta_l(\mathbf{S}_1, \ldots, \mathbf{S}_r(x)) = 0$, where $\Theta_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x)$ is a $(T + p + 1) \times m$ matrix

$$\Theta_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x) \triangleq \begin{pmatrix} -(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l) W_{ij})^T g(x) : (i,j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \cap \mathbf{S}_l \\ (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l) W_{ij})^T g(x) : (i,j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \setminus \mathbf{S}_l \\ -W(\mathbf{S}_l)^T g(x) \\ A \end{pmatrix} \tag{2.25}$$

and (e) $\forall l = 1, \ldots, r$, $y_l^T \Lambda_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x) < 0$, where $\Lambda_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x) \in \mathbb{R}^{T+p+1}$ is given by

$$\Lambda_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x) \triangleq \begin{pmatrix} (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l) W_{ij})^T f(x) : (i,j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \cap \mathbf{S}_l \\ -(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l) W_{ij})^T f(x) : (i,j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \setminus \mathbf{S}_l \\ \overline{W}(\mathbf{S}_l)^T f(x) \\ c \end{pmatrix} \tag{2.26}$$

Proof. The approach is to prove that condition (ii) of Proposition 2.2 holds; condition (i) holds automatically if each $\mathbf{S}_1, \ldots, \mathbf{S}_r$ satisfies condition (ii) of Lemma 2.6. We have that conditions (a) and (b) are equivalent to $b(x) = 0$ and (2.15). In order for x to be a safety counterexample, for all $l = 1, \ldots, r$, at least one of Eqs. (2.11) and (2.12) must fail. Equivalently, for all $l = 1, \ldots, r$, there does not exist u satisfying

$$\begin{aligned}
-(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T g(x)u &\leq (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T f(x) \ \forall (i,j) \in T(\mathbf{S}_1, \ldots, \mathbf{S}_r) \cap \mathbf{S}_l \\
-(\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T g(x)u &\geq (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_l)W_{ij})^T f(x) \ \forall (i,j) \in T(\mathbf{S}_1, \ldots, \mathbf{S}_r) \setminus \mathbf{S}_l \\
-\overline{W}_{ij}(\mathbf{S}_l)^T g(x)u &\leq \overline{W}(\mathbf{S}_l)^T f(x) \\
Au &\leq c
\end{aligned}$$

By Farkas Lemma, non-existence of such a u is equivalent to existence of y_l satisfying $y_l \geq 0$ as well as (2.25) and (2.26). □

The verification problem of Lemma 2.7 can be mapped to solving a nonlinear program

$$\begin{array}{ll}
\min_{x, y_1, \ldots, y_r} & \max_{l=1,\ldots,r} \left\{ y_l^T \Lambda_l(\mathbf{S}_1, \ldots, \mathbf{S}_r, x) \right\} \\
\text{s.t.} & (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_1)W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}_1) < 0 \ \forall (i,j) \notin S_1 \cup \cdots \cup S_r \\
& (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_1)W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}_1) > 0 \ \forall (i,j) \in S_1 \cap \cdots \cap S_r \\
& (\overline{\mathbf{W}}_{i-1}(\mathbf{S}_1)W_{ij})^T x + \overline{r}_{ij}(\mathbf{S}_1) = 0 \ \forall (i,j) \in \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r) \\
& y_l^T \Theta_l(\mathbf{S}_1, \ldots, \mathbf{S}_r(x)) = 0 \ \forall l = 1, \ldots, r \\
& y_l \geq 0 \ \forall l = 1, \ldots, r
\end{array} \tag{2.27}$$

and checking whether the optimal value is nonnegative (safe) or negative (unsafe). This nonlinear program is solved for each $(\mathbf{S}_1, \ldots, \mathbf{S}_r) \in \mathcal{V}$, where $\mathcal{V}$ is computed during the enumeration phase.

Note that, if $g(x)$ is constant, then the constraints of the nonlinear program are linear, and if $f(x)$ is linear then the problem reduces to solving a system of linear and bilinear inequalities. Furthermore, if $f(x)$ can be bounded over $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r)$ and $b(x) = 0$, then this bound can be used to derive a linear relaxation to the objective function of (2.27), thus relaxing the nonlinear program to a linear program.

The complexity of this approach can also be reduced by utilizing sufficient conditions for safety that are easier to check. For instance, if there exists $u \in \mathcal{U}$ such that $\overline{W}(\mathbf{S})^T(f(x)+g(x)u) \geq 0$

for all $\mathbf{S} \in \mathbf{S}(x)$, then the criteria of Lemma 2.7 are satisfied. In particular, if $\overline{W}(\mathbf{S})^T f(x) \geq 0$ for all $\mathbf{S} \in \mathbf{S}(x)$, then the conditions are satisfied with $u = 0$.

The following result is a straightforward consequence of Proposition 2.2, Lemma 2.6, and Lemma 2.7.

Theorem 2.3. *Suppose that the conditions of Lemma 2.6 are satisfied for each $\mathbf{S} \in \mathcal{S}$ and the conditions of Lemma 2.7 are satisfied for each $\{\mathbf{S}_1, \ldots, \mathbf{S}_r\} \in \mathcal{V}$. Then the NCBF b satisfies the conditions of Problem 2.1.*

Proof. Suppose that x is a safety counterexample for the NCBF b with $b(x) = 0$. If $x \in \text{int}\overline{\mathcal{X}}(\mathbf{S})$ for some $\mathbf{S}$, then we have that $\mathbf{S} \in \mathcal{S}$ and hence a contradiction with Lemma 2.6. If $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \overline{\mathcal{X}}(\mathbf{S}_r)$ for some $\mathbf{S}_1, \ldots, \mathbf{S}_r$, then there is a contradiction with Lemma 2.7. □

Theorem 2.3 implies that, if passing the verification, then any control policy consistent with barrier function b will be safe.

2.5.3 Efficient Verification

The efficient verification component takes the sets of boundary hyperplanes $\mathcal{S}$ and boundary hinges $\mathcal{V}$ and checks that the conditions of Proposition 2.2 hold for each of them. As pointed out after Proposition 2.2, the problem of searching for safety counterexamples can be decomposed into searching for correctness, hyperplane, and hinge counterexamples. In order to maximize the efficiency of our approach, we first consider the least computationally burdensome verification task, namely, searching for correctness counterexamples. We then search for hyperplane counterexamples, followed by hinge counterexamples.

Correctness Verification: The correctness condition ((2.19)) can be verified for boundary hyperplane $\overline{\mathcal{X}}(\mathbf{S})$ by solving the nonlinear program (2.28).

$$\begin{array}{ll} \min_x & h(x) \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{2.28}$$

When $h(x)$ is convex, (2.28) can be solved efficiently. Otherwise, dReal can be used to check satisfiability of (2.28) in a tractable runtime.

Hyperplane Verification: Hyperplane counterexamples can be identified by solving the optimization problem (2.29).

$$\begin{array}{ll} \min_x & \max\{\overline{W}(\mathbf{S})^T(f(x)+g(x)u) : u \in \mathcal{U}\} \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{2.29}$$

Solving (2.29) can be made more efficient when additional structures on the input set $\mathcal{U}$ and dynamics f and g are present. Consider a case $\mathcal{U} = \{D\omega : ||\omega||_\infty \leq 1\}$. The bounded input set $\mathcal{U}$ allows us to replace the maximization over u with L-1 norm term $\|\overline{W}(\mathbf{S})^T g(x)D\|_1$, which simplifies the computational complexity. In this case, the problem reduces to the nonlinear program

$$\begin{array}{ll} \min\limits_x & \overline{W}(\mathbf{S})^T f(x) + \|\overline{W}(\mathbf{S})^T g(x)D\|_1 \\ \text{s.t.} & \overline{W}(\mathbf{S})^T x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{2.30}$$

If $\mathcal{U} = \mathbb{R}^m$, then by Corollary 2.1, the problem can be reduced to the nonlinear program

$$\begin{array}{ll} \min_x & \overline{W}(\mathbf{S})^T f(x) \\ \text{s.t.} & \overline{W}(\mathbf{S})^T g(x) = 0, \overline{W}(\mathbf{S})x + \overline{r}(\mathbf{S}) = 0, x \in \overline{\mathcal{X}}(\mathbf{S}) \end{array} \tag{2.31}$$

If $f(x)$ and $g(x)$ are linear in x, then the problem boils down to a linear program. If bounds on the Lipschitz coefficients of f and g are available, then they can be used to derive approximations of (2.30).

If $g(x)$ is a constant matrix G, then safety is automatically guaranteed if $\overline{W}(\mathbf{S})^T G \neq 0$. If $f(x)$ is linear in x as well, then (2.31) is a linear program.

Hinge Verification: The hinge $\mathbf{V} = \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ can be certified by solving the nonlinear optimization problem (2.32).

$$\begin{array}{ll} \min_x & \max\{\overline{W}(\mathbf{S}_l)^T(f(x)+g(x)u) : l = 1, \ldots, r, u \text{ satisfies (2.16)–(2.17)}\} \\ \text{s.t.} & x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r), \overline{W}(\mathbf{S}_1)^T x + \overline{r}(\mathbf{S}_1) = 0 \end{array} \tag{2.32}$$

In practice, simple heuristics are often sufficient to verify safety of hinges without resorting to solving (2.32). If $\overline{W}(\mathbf{S})^T f(x) > 0$ for all $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \cap \overline{\mathcal{X}}(\mathbf{S}_r)$, then the control input $u = \mathbf{0}$ suffices to ensure safety. Furthermore, if $\mathcal{U} = \mathbb{R}^m$ and there exists $i \in \{1, \ldots, m\}$ and $s \in \{0, 1\}$ such that $\text{sign}((\overline{W}(\mathbf{S}_l)^T g(x))_i) = s$ for all $x \in \overline{\mathcal{X}}(\mathbf{S}_1) \cap \cdots \overline{\mathcal{X}}(\mathbf{S}_r)$ and $l = 1, \ldots, r$, then u can be chosen as $u_i = Ks$ for some sufficiently large $K > 0$ to ensure that the conditions of Proposition 2.2.

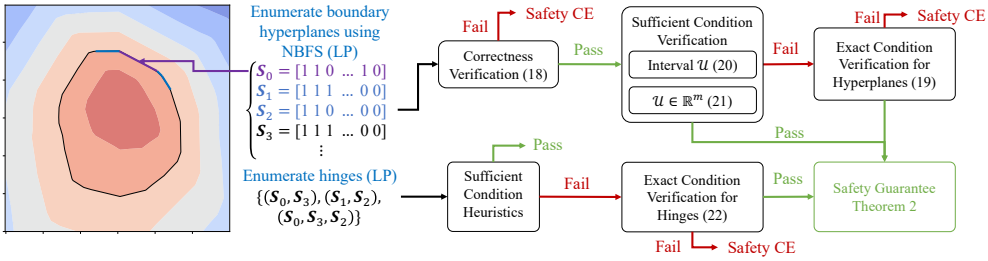


Figure 2.3: Overview of the Efficient Exact Verifier for ReLU NCBFs

EEV decomposes the NCBF into hyperplanes and hinges and verifies each component hierarchically, with novel tractable sufficient conditions verified first and the more complex exact conditions checked only if the sufficient conditions fail. Given an NCBF $b_\theta(x)$, the verification of EEV returns (i) a Boolean variable that is 'true' if the conditions of Proposition 2.2 are satisfied and 'false' otherwise, and (ii) a safety counterexample $\hat{x}$ that violates the conditions of Proposition 2.2 if the result is 'false'. Algorithm 4 presents an overview of the verification of EEV. The algorithm consists of an enumeration stage to identify all boundary hyperplanes and hinges, and a verification stage to certify satisfaction of conditions in Proposition 2.2 for all hyperplanes and hinges.

Safety Guarantee: The safety guarantees of our proposed approach are summarized in Theorem 2.4.

Theorem 2.4. *Given a NCBF $b_\theta(x)$, $b_\theta(x)$ is a valid NCBF if it passes the verification of Algorithm 4 using dReal to solve the optimization problems* (2.28), (2.29), *and* (2.32).

Proof. The proof is derived from completeness of the enumeration in Algorithm 4 and dReal. By Lemma 2.5, the initial boundary activation set $\mathbf{S}_0$ is ensured to be identified by the EEV in Line 4 Algorithm. 4. Given $\mathbf{S}_0$ and the enumeration in Line 5 and 6 Algorithm. 4 being complete, the completeness of $\mathcal{S}$ and $\mathcal{V}$ is guaranteed by Proposition 2.3. By dReal solving

Algorithm 4 Efficient Exact Verification

Input: $n, \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$
Output: Verification Boolean result r, Categorized counterexample $\hat{x}_{ce}$
procedure EFFICIENT EXACT VERIFICATION($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$)
    $\mathbf{S}_0 \leftarrow$ ENUMINIT($\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}, \mathcal{T}_{\mathcal{I}}$) ▷ Initial Activation Set Identification, Section 2.4.2
    $\mathcal{S} \leftarrow$ NBFS($\mathbf{S}_0$) ▷ Activation Sets Enumeration, Section 2.4.2
    $r, \hat{x}_{ce}^{(c)} \leftarrow$ CorrectnessVerifier($\mathcal{S}$) ▷ Correctness Verification, Section 2.5.3
    $r, \hat{x}_{ce}^{(h)} \leftarrow$ HyperplaneVerifier($\mathcal{S}$) ▷ Hyperplane Verification, Section 2.5.3
    $\mathcal{V} \leftarrow$ HingeEnum($\mathcal{S}, n$) ▷ Hinges Enumeration, Section 2.4.2
    $r, \hat{x}_{ce}^{(g)} \leftarrow$ HingeVerifier($\mathcal{V}$) ▷ Hinge Verification, Section 2.5.3
    Return r, $\hat{x}_{ce}$
end procedure

the equivalent NLPs, the conditions of Proposition 2.2, are satisfied. Therefore, $b_\theta(x)$ is a valid NCBF □

2.6 Experiment

2.6.1 Experiment Settings

In this section, we evaluate our proposed method to verify neural control barrier functions on three systems, namely Darboux, obstacle avoidance and spacecraft rendezvous.

Darboux

We consider the Darboux system [96], a nonlinear open-loop polynomial system that has been widely used as a benchmark for constructing barrier certificates. The dynamic model is given in the supplement. We obtain the trained NCBF by following the method proposed in [69]. We show the settings of NCBF verification for Darboux system whose dynamic is defined as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 + 2x_1x_2 \\ -x_1 + 2x_1^2 - x_2^2 \end{bmatrix}. \tag{2.33}$$

We define state space, initial region, and unsafe region as $\mathcal{X} : \{\mathbf{x} \in \mathbb{R}^2 : x \in [-2, 2] \times [-2, 2]\}$, $\mathcal{X}_I : \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 1 \leq x_2 \leq 2\}$ and $\mathbf{x}_U : \{\mathbf{x} \in \mathbb{R}^2 : x_1 + x_2^2 \leq 0\}$ respectively.

Obstacle Avoidance

We evaluate our proposed method on a controlled system [97]. We consider an Unmanned Aerial Vehicles (UAVs) avoiding collision with a tree trunk. We model the system as a Dubins-style [98] aircraft model. The system state consists of 2-D position and aircraft yaw rate $x := [x_1, x_2, \psi]^T$. We let u denote the control input to manipulate yaw rate and the dynamics defined in the supplement. We train the NCBF via the method proposed in [69] with v assumed to be 1 and the control law u designed as $u = \mu_{nom}(x) = -\sin\psi + 3 \cdot \frac{x_1 \cdot \sin\psi + x_2 \cdot \cos\psi}{0.5 + x_1^2 + x_2^2}$. We next evaluate that our proposed method on a controlled system [97]. The system state consists of 2-D position and aircraft yaw rate $x := [x_1, x_2, \psi]^T$. We let u denote the control input to manipulate yaw rate and define the dynamics as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \sin\psi \\ v \cos\psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix}. \tag{2.34}$$

We define the state space, initial region and unsafe region as $\mathcal{X}$, $\mathcal{X}_I$ and $\mathcal{X}_U$, respectively as

$$\begin{aligned} \mathcal{X} &: \{\mathbf{x} \in \mathbb{R}^3 : x_1, x_2, \psi \in [-2, 2] \times [-2, 2] \times [-2, 2]\} \\ \mathcal{X}_I &: \{\mathbf{x} \in \mathbb{R}^3 : -0.1 \leq x_1 \leq 0.1, -2 \leq x_2 \leq -1.8,\ -\pi/6 < \psi < \pi/6\} \\ \mathcal{X}_U &: \{\mathbf{x} \in \mathbb{R}^3 : x_1^2 + x_2^2 \leq 0.04\} \end{aligned} \tag{2.35}$$

Spacecraft Rendezvous

We evaluate our approach on a spacecraft rendezvous problem from [99]. A station-keeping controller is required to keep the "chaser" satellite within a certain relative distance to the "target" satellite. The state of the chaser is expressed relative to the target using linearized Clohessy–Wiltshire–Hill equations, with state $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$, control input $u = [u_x, u_y, u_z]^T$ and dynamics defined in the supplement. We train the NCBF as in [16]. The state of the chaser is expressed relative to the target using linearized Clohessy–Wiltshire–Hill equations, with state $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$, control input $u = [u_x, u_y, u_z]^T$ and dynamics

defined as follows.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}. \tag{2.36}$$

We define the state space and unsafe region as $\mathcal{X}$ and $\mathcal{X}_U$, respectively as

$$\begin{aligned} \mathcal{X} &: \left\{ \mathbf{x} \in \mathbb{R}^3 : p, v, \in [-1.5, 1.5] \times [-1.5, 1.5] \right\} \\ \mathcal{X}_U &: \left\{ 0.25 \leq r \leq 1.5, \text{ where } r = \sqrt{p_x^2 + p_y^2 + p_z^2} \right\} \end{aligned} \tag{2.37}$$

We obtain the trained NCBF with neural CLBF training in [16] with a nominal model predictive controller.

hi-ord$_8$

We evaluate our approach on an eight-dimensional system that first appeared in [40] to evaluate the scalability of proposed verification method. The dynamic model of the system is captured by an ODE as follows.

$$x^{(8)} + 20x^{(7)} + 170x^{(6)} + 800x^{(5)} + 2273x^{(4)} + 3980x^{(3)} + 4180x^{(2)} + 2400x^{(1)} + 576 = 0 \tag{2.38}$$

where we denote the i-th derivative of variable x by $x^{(i)}$. We define the state space and unsafe region as $\mathcal{X}$ and $\mathcal{X}_U$, respectively as

$$\begin{aligned} \mathcal{X} &: \left\{ x_1^2 + \ldots + x_8^2 \leq 4 \right\} \\ \mathcal{X}_U &: \left\{ (x_1 + 2)^2 + \ldots + (x_8 + 2)^2 \leq 0.16 \right\} \end{aligned} \tag{2.39}$$

We obtain the trained NCBFs with training method proposed in [40].

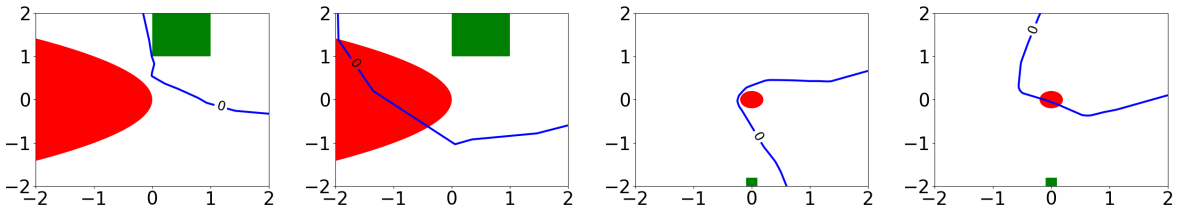


Figure 2.4: Comparison of NCBFs that pass and fail the proposed verification. We show 0-level set boundary in blue, initial region in green and the unsafe region in red. (a) and (b) visualize NCBFs for Darboux. (c) and (d) shows projection of NCBFs for Obstacle Avoidance.

2.6.2 LiRPA-based Verification Results

We first verify that $\mathcal{D} \subseteq \mathcal{C}$ in the Darboux and obstacle avoidance scenarios. We trained four 1 hidden layer NCBFs with 20, 5, 32, 10 neurons, respectively. NCBF (a) and (c) completed training, (b) terminated after 3 epochs and (d) terminated after 5 epochs. Our verification algorithm identifies safety violations. NCBF (a) and (c) pass the verification while NCBF (b) and (d) fail. As shown in Fig. 2.4a, the fully-trained NCBF (a) separates the safe and unsafe regions while the NCBF (b) with unfinished training has a boundary that intersects the unsafe region in Fig. 2.4b. As shown in Fig. 2.4d, the 2-D projection with $\psi = -0.5$ of NCBF (d) shows its boundary overlap the unsafe region.

We next verify the positive invariance of $\mathcal{D}$. The experiments run on a 64-bit Windows PC with Intel i7-12700 processor, 32GB RAM and NVIDIA GeForce RTX 3080 GPU. All NCBFs pass feasibility verification with run-time presented in Table 2.1. We compare the run-time of proposed verification with SMT-based approaches using translator proposed in [40] and SMT solver dReal [100] and Z3 [101]. The proposed method can verify NCBFs with more ReLU neurons while SMT-based verifier return nothing. All of the generated NCBFs for obstacle avoidance and spacecraft rendezvous passed the verification. The run-times are presented in Table 2.2. We find that the activation set sizes grow with the size of neural network and the state dimension n. We conjecture that this growth rate could be reduced by using tighter approximations for the activated sets.

A key feature of our approach is that the verification process does not depend on the control policy μ, but rather we verify that any μ satisfying (2.11)–(2.13) for all $x \in \partial\mathcal{D}$ is safe. This improves flexibility while reducing false negatives in safety verification, as we illustrate in

Table 2.1: Comparison of verification run-time of NCBF in seconds. The table contains the dimension n, network architecture with σ denoting ReLU, the number of activation sets N and run-time of proposed method including time of enumerating, verification and total run-time denoted as t_e, t_v and T, respectively. We compare with the run-time of dReal (T_{dReal}) and Z3 (T_{Z3}).

Case	n	NN Architecture	N	t_e	t_v	T	T_{dReal}	T_{Z3}
Darboux	2	2-20-σ-1	13	0.39	0.04	0.43	0.36	>3hrs
	2	2-32-σ-1	20	0.23	0.03	0.27	1.27	>3hrs
	2	2-64-σ-1	47	1.89	0.13	2.02	>3hrs	>3hrs
	2	2-96-σ-1	63	4.17	0.14	4.31	>3hrs	>3hrs
	2	2-128-σ-1	65	4.77	0.03	4.90	>3hrs	>3hrs
	2	2-512-σ-1	258	33.44	0.61	34.05	>3hrs	>3hrs
	2	2-1024-σ-1	548	107.63	0.81	108.44	>3hrs	>3hrs
Darboux	2	2-10-σ-10-σ-1	4	0.42	0.05	0.47	7.70	15.81
	2	2-16-σ-16-σ-1	26	2.13	0.11	2.23	>3hrs	>3hrs
	2	2-32-σ-32-σ-1	58	8.35	6.28	14.64	>3hrs	>3hrs
	2	2-48-σ-48-σ-1	58	13.69	0.19	13.88	>3hrs	>3hrs
	2	2-64-σ-64-σ-1	102	31.30	0.47	31.77	>3hrs	>3hrs
	2	2-256-σ-256-σ-1	402	319.81	1.92	321.73	>3hrs	>3hrs
	2	2-512-σ-512-σ-1	1150	619.10	0.75	619.85	>3hrs	>3hrs
hi-ord$_8$	8	8-8-σ-1	98	3.70	0.02	3.72	>3hrs	>3hrs
	8	8-16-σ-1	37	35.05	0.0	35.05	>3hrs	>3hrs

the following example. The NCBF for obstacle avoidance with one hidden layer and 32 neurons fails the SMT-based verification using dReal given nominal controller μ_{nom}. The counter example is at the point $(-0.20, 0, -0.73)$. At this point, we have $b(x) = 0.0033$, $\frac{\partial b}{\partial x} f(x) = 0.0095$, which satisfies the sufficient conditions for safety in Lemma 2.4. However, the nominal controller yields $\frac{\partial b}{\partial x}(f(x) + g(x)u) = -1.30$ and hence fails verification. Hence, by following a safe control policy of the form (2.14) with the learned value of b and nominal policy μ_{nom}, we can retain the performance of μ_{nom} while still ensuring safety.

Table 2.2: Comparison of verification run-time of NCBF in seconds. We denote the run-time as 'UTD' when the method is unable to be directly used for verification.

Case	n	NN Architecture	N	t_e	t_v	T	T_{dReal}	T_{Z3}
OA	3	3-32-σ-1	436	6.94	0.40	7.35	>6hrs	>6hrs
	3	3-64-σ-1	1645	19.17	1.87	21.05	>6hrs	>6hrs
	3	3-96-σ-1	3943	58.96	6.26	65.22	>6hrs	>6hrs
	3	3-128-σ-1	5695	169.51	15.56	185.07	>6hrs	>6hrs
OA	3	3-16-σ-16-σ-1	467	18.72	2.66	21.39	>6hrs	>6hrs
	3	3-32-σ-32-σ-1	1324	218.86	54.51	273.37	>6hrs	>6hrs
	3	3-48-σ-48-σ-1	3754	3197.59	9.75	3207.34	>6hrs	>6hrs
	3	3-64-σ-64-σ-1	6545	11730.28	18.22	11748.50	>6hrs	>6hrs
SR	6	6-8-σ-8-σ-8-1	270	78.77	10.23	89.00	UTD	UTD
	6	6-16-σ-16-σ-16-1	32937	1261.60	501.06	1762.67	UTD	UTD
	6	6-32-σ-32-σ-32-1	207405	12108.11	1798.08	13906.19	UTD	UTD

The computational complexity of our approach will be determined by several factors including the dimension of the state, the number of layers, the number of neurons in each layer, and the geometry of the 0-level set of the NCBF. As shown in Table 2.2, the dimension of the system plays the most important role.

2.6.3 Exact Efficient Verification Results

The results presented in Table 2.3 illustrate a significant improvement in verification efficiency for Neural Control Barrier Functions NCBFs using the proposed method. In the table t_h represents the time spent searching for hyperplanes containing the CBF boundary and verifying CBF sufficient conditions on these boundaries, and t_g represents the time spent in hinge enumeration and verification. The total time, T, is the sum of t_h and t_g. We compare

Table 2.3: Comparison of verification run-time of NCBF in seconds. We denote the run-time as 'UTD' when the method is unable to be directly used for verification.

Case	n	L	M	N	t_h	t_g	SEEV	Baseline [39]	dReal	Z3
Darboux	2	2	256	15	2.5s	0	2.5s	315s	>3h	>3h
	2	2	512	15	3.3s	0	3.3s	631s	>3h	>3h
OA	3	2	16	86	0.41s	0	0.41s	16.0s	>3h	>3h
	3	4	8	15	0.39	0	0.39	16.1s	>3h	>3h
	3	4	16	136	0.65s	0	0.65s	36.7s	>3h	>3h
	3	1	128	5778	20.6s	0	20.6s	207s	>3h	>3h
hi-ord$_8$	8	2	8	73	0.54s	0	0.54s	>3h	>3h	>3h
	8	2	16	3048	11.8s	0	11.8s	>3h	>3h	>3h
	8	4	16	3984	22.4s	0	22.4s	>3h	>3h	>3h
SR	6	2	8	2200	7.1s	2.7s	9.8s	179s	UTD	UTD
	6	4	8	4918	45.8s	14.3s	60.1s	298.7s	UTD	UTD

our approach with three baselines, exact verification [39], SMT-based verification [40] with dReal and Z3. Baseline methods' run times are represented by Baseline [39], dReal, and Z3.

In the Darboux cases, our method achieves verification in 2.5 seconds and 3.3 seconds for $M = 256$ and $M = 512$ respectively, whereas baseline methods take substantially longer, with Baseline [39] taking 315 seconds and 631 seconds, and both dReal and Z3 taking more than 3 hours. Similarly, in the OA cases, our method's run times range from 0.39 seconds to 20.6 seconds, faster than the baseline methods. In the more higher dimensional systems high-ord$_8$ and SR, our method significantly outperforms Baseline [39]. Specifically, in high-ord$_8$ our methods finishes within 22.4 seconds while Baseline [39], dReal and Z3 times out, due to the need to enumerate the 8-dimensional input space. For the SR case, SEEV's run time are 9.8 seconds and 60.1 seconds, beating Baseline [39] which takes 179 seconds and 298.7 seconds respectively. Neural barrier certificate based dReal and Z3 are able to directly applicable since they require an explicit expression of the controlled feedback system. However, the SR system is manipulated by an NCBF-based safe controller that is nontrivial to derive an explicit expression.

Note that hinge enumeration and certification may be time-consuming procedure, since they involve enumerating all combinations of hyperplanes. However, the results from Table 2.3 show that the certification can be completed on most hyperplanes with sufficient condition verification in Section 2.5.3, greatly improving the overall run time.

2.7 NCBF Synthesis with Efficient Exact Verification

In this section, we present the framework to synthesize NCBF $b_\theta(x)$ to ensure the safety of the system (2.1). The synthesis framework aims to train an NCBF and construct an NCBF-based safe control policy. We first formulate the problem and present an overview of the framework in 2.7.1. Then we demonstrate the design of the loss function in 2.7.2.

2.7.1 Overall Formulation

Our primary objective is to synthesize a ReLU Neural Control Barrier Function (ReLU-NCBF) for (2.1) and develop a safe control policy to ensure system safety.

Problem 2.2. *Given a system* (2.1)*, initial set $\mathcal{I}$ and a safety set $\mathcal{C}$, synthesize a ReLU NCBF $b_\theta(x)$ parameterized by θ such that the conditions in Proposition 2.2 are satisfied. Then construct a control policy to synthesize u_t such that the system remains positive invariant in $\mathcal{D} := \{x : b_\theta(x) \geq 0\} \subseteq \mathcal{C}$.*

We propose Synthesis with Efficient Exact Verification (SEEV) to address this problem with the synthesis framework demonstrated in Fig. 2.5. The training dataset $\mathcal{T}$ is initialized by uniform sampling over $\mathcal{X}$. The training framework consists of two loops. The inner loop attempts to choose parameter θ for $b_\theta(x)$ to satisfy the safety condition by minimizing the loss function over training data $\mathcal{T}$. The outer loop validates a given NCBF $b_\theta(x)$ by searching for safety counterexamples $\hat{x}_{ce}$ and updates the training dataset as $\mathcal{T} \cup \{\hat{x}_{ce}\}$.

To train the parameters of the NCBF to satisfy the conditions of Proposition 2.2, we propose a loss function that penalizes the NCBF for violating constraints (i) and (ii) at a collection of sample points. The loss function is a weighted sum of three terms. The first term is the correctness loss penalizing state $\hat{x} \in \mathcal{X} \setminus \mathcal{C}$ with $b_\theta(x) \geq 0$. The second term is verification loss that penalizes states $\hat{x}$ that $\nexists u$ such that (2.16)-(2.18) hold. The third term is a regularizer minimizing the number of hyperplanes and hinges along the boundary. However, minimizing the loss function is insufficient to ensure safety [89] because there may exist safety counterexamples outside of the training dataset. In order to guarantee safety, SEEV introduces an efficient exact verifier to certify whether $b_\theta(x)$ meets the safety conditions

outlined in Proposition 2.2. The verifier either produces a proof of safety or generates a safety counterexample that can be added to the training dataset to improve the NCBF.

The integration of the verifier can improve safety by adding counterexamples to guide the training process, however, it may also introduce additional computation complexity.

We propose a combined approach, leveraging two complementary methods to address this issue. First, the verification of SEEV introduces an efficient algorithm in Section 2.4 to mitigate the computational scalability challenges that arise as neural network depth and dimensionality increase. Second, SEEV introduces a regularizer to limit the number of boundary hyperplanes and hinges to be verified, addressing the complexity that arises as neural network size increases.

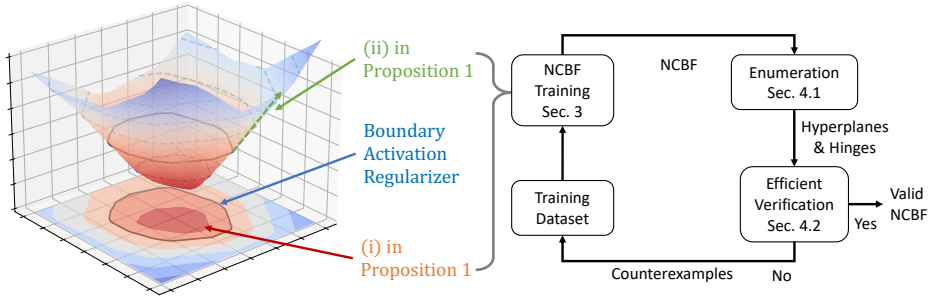


Figure 2.5: SEEV: Synthesis with Efficient Exact Verifier for ReLU NCBF

2.7.2 Loss Function Design and NCBF Training

The goal of the inner loop is to choose parameters θ so that the conditions of Proposition 2.2 are satisfied for all $\hat{x} \in \mathcal{T}$ and the computational cost of verifying safety is minimized. To achieve the latter objective, we observe (based on results presented in Table 2.4) that the computational complexity of verification grows with the cardinality of the collection of activation sets that intersect the safety boundary $\partial\mathcal{D}$. The collection is defined as $\mathcal{B} := \{\mathbf{S}_i : \partial\mathcal{D} \cap \overline{\mathcal{X}}(\mathbf{S}_i) \neq \emptyset\}$. Hence, we formulate the following unconstrained optimization problem to search for θ.

$$\min_{\theta} \quad \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}}(\mathcal{T}) + \lambda_f\mathcal{L}_f(\mathcal{T}) + \lambda_c\mathcal{L}_c(\mathcal{T}) \tag{2.40}$$

where $\mathcal{L}_{\mathcal{B}}(\mathcal{T})$ regularizer to minimize $|\mathcal{B}|$, $\mathcal{L}_f(\mathcal{T})$ is the loss penalizing the violations of constraint (2.16)-(2.18) ((i) of Proposition 2.2), $\mathcal{L}_c(\mathcal{T})$ penalizes the violations of constraint (2.19) ((ii) of Proposition 2.2), and $\lambda_{\mathcal{B}}$, λ_f and λ_c are non-negative coefficients defined as follows.

$\mathcal{L}_f$ Regularizer: For each sample $\hat{x} \in \mathcal{T}$ the safe control signal is calculated by

$$\min_{u,r} \quad ||u - \pi_{nom}(x)||_2^2 \qquad s.t. \quad \mathcal{W}(\mathbf{S}_l)^T(f(\hat{x}) + g(\hat{x})u) + r \geq 0 \tag{2.41}$$

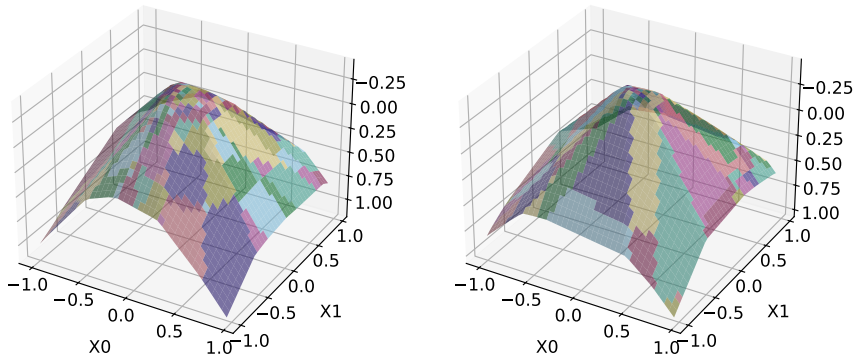
where $\mathcal{W} = \overline{W}$ for differentiable points and $\mathcal{W} = \tilde{W}$ defined as the subgradient at non-differentiable points. The regularizer $\mathcal{L}_f$ enforces the satisfaction of the constraint by inserting a positive relaxation term r in the constraint and minimizing r with a large penalization in the objective function. We have the loss $\mathcal{L}_f$ defined as $\mathcal{L}_f = ||u - \pi_{nom}(x)||_2^2 + r$. We use [102] to make this procedure differentiable, allowing us to employ the relaxation loss into the NCBF loss function design.

$\mathcal{L}_c$ Regularizer: $\mathcal{L}_c$ regularizer enforces the correctness of the NCBF. In particular, it enforces the $b_\theta(x)$ of safe samples $x \in \mathcal{T}_{\mathcal{I}}$ to be positive, and $b_\theta(x)$ of unsafe samples $\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}$ to be negative. Define $N_{\text{safe}} = |\mathcal{T}_{\mathcal{I}}|$ and $N_{\text{unsafe}} = |\mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}|$, and with a small positive tuning parameter $\epsilon > 0$, the loss term $\mathcal{L}_c$ can be defined as

$$\mathcal{L}_c = a_1 \frac{1}{N_{\text{safe}}} \sum_{x \in \mathcal{T}_{\mathcal{I}}} [\epsilon - b_\theta(x)]_+ + a_2 \frac{1}{N_{\text{unsafe}}} \sum_{x \in \mathcal{T}_{\mathcal{X}\setminus\mathcal{C}}} [\epsilon + b_\theta(x)]_+ \tag{2.42}$$

where $[\cdot]_+ = max(\cdot, 0)$. a_1 and a_2 are positive parameters controlling penalization strength of the violations of safe and unsafe samples.

$\mathcal{L}_{\mathcal{B}}$ Regularizer: We propose a novel regularizer to limit the number of boundary hyperplanes and hinges by penalizing the dissimilarity, i.e., $\mathbf{S}(\hat{x}_i)\Delta\mathbf{S}(\hat{x}_j)$ of boundary activation sets $\mathbf{S}(\hat{x}) \in \mathcal{B}$. However, the dissimilarity measure of boundary activation sets is inherently nondifferentiable. To address this issue the regularizer introduces the generalized sigmoid function $\sigma_k(z) = \frac{1}{1+\exp(-k \cdot z)}$ to compute the vector of smoothed activation defined as $\phi_{\sigma_k}(x) := [\sigma_k(z_{i,j}), \forall i, j \in \{1, \ldots, L\} \times \{1, \ldots, M_i\}]$. The $\mathcal{L}_{\mathcal{B}}$ regularizer conducts the following two steps to penalize dissimilarity.



(a) $r = 0$ (b) $r = 50$

Figure 2.6: Effects of boundary regularization (r) on activation sets along the boundary. The figures show the results from a neural network with 4 layers of 8 hidden units, applied to the Spacecraft case. The surface represents the first two dimensions with the last four dimensions fixed at 0. Increasing r results in more organized boundary activation sets.

In the first step, the regularizer identifies the training data in the boundary hyperplanes and hinges denoted as $\hat{x} \in \mathcal{T}_{\partial\mathcal{D}}$. The set is defined as $\mathcal{T}_{\partial\mathcal{D}} := \{\hat{x} : \hat{x} \in \overline{\mathcal{X}}(\mathbf{S}), \forall \mathbf{S} \in \mathcal{B}\}$. To further improve the efficiency, the regularizer approximates $\mathcal{T}_{\partial\mathcal{D}}$ with a range-based threshold ϵ on the output of the NCBF, i.e., $|b_\theta(\hat{x})| \leq \epsilon$.

The second step is to penalize the dissimilarity of $\mathbf{S} \in \mathcal{B}$. To avoid the potential pitfalls of enforcing similarity across the entire boundary $\mathcal{B}$, the regularizer employs an unsupervised learning approach to group the training data into $N_{\mathbf{B}}$ clusters. We define the collection of the activation set in each cluster as $\mathbf{B}_i \subseteq \mathcal{B}$ and the collection in each cluster as $\mathcal{T}_{\mathbf{B}_i} := \{\hat{x} : \hat{x} \in \bigcup_{\mathbf{S}\in\mathbf{B}_i} \overline{\mathcal{X}}(\mathbf{S})\}$. The $\mathcal{L}_{\mathcal{B}}$ is then defined as follows, with an inner sum over all pairs of $\hat{x} \in \mathcal{T}_{\mathbf{B}_i}$ and an outer sum over all clusters.

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{N_{\mathbf{B}}} \sum_{\mathbf{B}_i \in \mathcal{B}} \frac{1}{|\mathcal{T}_{\mathbf{B}_i}|^2} \sum_{\hat{x}_i, \hat{x}_j \in \mathcal{T}_{\mathbf{B}_i}} \|\phi_{\sigma_k}(x_i) - \phi_{\sigma_k}(x_j)\|_2^2, \tag{2.43}$$

2.7.3 SEEV Evaluation

Table 2.4 and Figure 2.6 illustrates the impact of regularization on the CBF boundary's activation sets. Table 2.4 compares various configurations, where n denotes the input dimensions, L represents the number of layers, and M indicates the number of hidden units per layer. N_o and N_r are the number of hyperplanes along the zero-level boundary of the CBF without and with regularization, respectively, with r indicating the regularization

Table 2.4: Comparison of N the number of boundary hyperplanes and C coverage of the safe region $\mathcal{D}$ of NCBF trained with and without boundary hyperplane regularizer denoted with subscripts $_r$ and $_o$.

Case	n	L	M	N_o	C_o	$N_{r=1}$	$\rho_{r=1}$	$N_{r=10}$	$\rho_{r=10}$	$N_{r=50}$	$\rho_{r=50}$
OA	3	2	8	26	89.46%	25	0.996	23.3	0.994	**13.3**	1.006
	3	2	16	116	83.74%	119	1.012	111	1.005	**98**	1.055
	3	4	8	40	91.94%	38	0.988	36	0.993	**13**	0.937
	3	4	16	156	87.81%	170	0.971	147	1.003	**64**	1.038
SR	6	2	8	2868	98.58%	2753	1	1559	1	**418**	1
	6	4	8	6371	98.64%	6218	1	3055	1	**627**	1
	6	2	16	N/A	N/A	204175	N/A	68783	N/A	**13930**	N/A

strength. C_o captures the CBF's coverage of the safe region, while $\rho_{(\cdot)} = C_{(\cdot)}/C_o$ represents the safety coverage ratio relative to the unregularized CBF. Notably, "N/A" entries indicate configurations where training a fully verifiable network was infeasible due to the excessive number of boundary hyperplanes, which leads the verification process to time out.

The results demonstrate that regularization effectively reduces the number of activation sets along the CBF boundary without compromising the coverage of the safe region. The efficiency is especially improved in cases with a greater number of hidden layers, where the unregularized model results in a significantly higher number of hyperplanes. For instance, in the SR case with $n = 6$, $L = 4$, and $M = 8$, the regularization reduces $N_{r=50}$ to 627 from $N_o = 6218$, maintaining the same safety coverage ($\rho_{r=50} = 1$). See Appendix 2.7.3 for hyperparameter sensitivity analysis.

Figure 2.6 illustrates the level sets of two CBFs trained for the SR case with $n = 6$, $L = 4$, and $M = 8$. These level sets are extracted from the first two dimensions with the rest set to zero. Each colored patch represents an activation pattern. The regularizer organizes the activation sets around the boundary, reducing unnecessary rapid changes and thereby enhancing the verification efficiency.

The experimental results presented in Table 2.5 demonstrate the effectiveness of Counter Example (CE) guided training on Darboux and hi-ord$_8$ system. In this method, after each training epoch, we calculate the Control Barrier Function (CBF) outputs on representative samples. If the CBF correctly categorizes the samples into safe and unsafe regions, the

certification procedure is initiated. If the CBF fails certification, the counter example is added to the training dataset for retraining. Otherwise, training is stopped early.

We capped the maximum training epochs at 50 and conducted three rounds of training for each network structure and system using different random seeds. The results indicate that without CE, the training process could basrely generate a CBF that passes certification. In contrast, with CE enabled, there was a success rate of at least 1/3 for most network structure, with verifiable policies generated in as few as 10 epochs. This highlights the improvement in training efficiency and reliability with the incorporation of CEs.

Case	L	M	No CE	With CE	
			sr	sr	min epoch
Darboux	2	8	0/3	3/3	38
	2	16	0/3	1/3	10
	4	8	0/3	1/3	43
	4	16	0/3	2/3	26
hi-ord$_8$	2	8	1/3	1/3	15
	2	16	0/3	2/3	19
	4	8	0/3	0/3	-
	4	16	0/3	2/3	13

Table 2.5: Success rates (sr) and minimum epochs required for certification with and without Counter Example (CE) guided training for different network structures on Darboux and hi-ord$_8$ systems.

Hyperparameters

Table 2.6 shows values the following hyperparameters used during CBF synthesis:

- N_{data}: number of samples to train CBF on.
- a_1: weight penalizing incorrect classification of safe samples in Equation 2.42.
- a_2: weight penalizing incorrect classification of unsafe samples Equation 2.42.
- λ_f: weight penalizing violation of Lie derivative condition of CBF in Equation 2.40.
- λ_c: weight penalizing correct loss for in Equation 2.40.
- n_{cluster}: number of clusters in $\mathcal{L}_{\mathcal{B}}$ regularization.

- k_σ: value of k used in generalized sigmoid function to perform differentiable activation pattern approximation.
- $\epsilon_{\text{boundary}}$: the threshold for range-based approximation of CBF boundary.

Case	N_{data}	a_1	a_2	λ_f	λ_c	n_{cluster}	k_σ	$\epsilon_{\text{boundary}}$
Darboux	5000	100	100	4.0	1.0	N/A	N/A	N/A
hi-ord$_8$	50000	100	200	1.0	1.0	N/A	N/A	N/A
OA	10000	100	100	2.0	1.0	5	4	1.0
SR	10000	100	100	2.0	1.0	5	4	1.0

Table 2.6: Hyperparameters of CBF synthesis.

Sensitivity Analysis of Hyperparameters

λ_c	SR	ME	N
1	0/3	x	x
10	0/3	x	x
100	**3/3**	**17**	**3265**
200	3/3	17.7	2922

(a) Ablation study on λ_c

λ_f	SR	ME	N
1	3/3	16.3	3254
2	**3/3**	**17**	**3265**
4	3/3	17	3352
8	2/3	29.5	3419.5

(b) Ablation study on λ_f

k	SR	ME	N
1	3/3	18	3842
2	3/3	15.7	3523
4	**3/3**	**17**	**3265**
8	1/3	10	1984

(c) Ablation study on k

Table 2.7: Ablation study for training hyperparameters. In each table, the bold lines indicate the baseline setting. **SR**: the success rate among runs with three random seeds. **ME**: the average first training epoch when a valid NCBF is obtained. **N**: the average number of boundary hyperplanes.

Next, we performed a sensitivity analysis of the hyperparameters. We chose the case study of Spacecraft Rendezvous with the number of layers $L = 4$ and the number of hidden units per layer $N = 8$. We studied the sensitivity of the training performance to the hyperparameters $\lambda_{\mathcal{B}}$, λ_f, and λ_c in Equation 8, corresponding to the weightings for regularizing the number of boundary hyperplanes, NCBF value violation, and NCBF Lie derivative violation, respectively. We also studied the sensitivity to the hyperparameter k employed in the modified sigmoid function $\sigma_k(z) = \frac{1}{1+\exp(-k\cdot z)}$ to approximate the regularization pattern. We compared against the settings: $\lambda_{\mathcal{B}} = 10$, $\lambda_f = 2$, $\lambda_c = 100$, and $k = 4$. For each hyperparameter, we chose four values to perform the ablation study: $\lambda_{\mathcal{B}} \in \{0, 1, 10, 50\}$, $\lambda_f \in \{1, 2, 4, 8\}$, $\lambda_c \in \{1, 10, 100, 200\}$, and $k \in \{1, 2, 4, 8\}$. For each setting, we performed three runs with different random seeds. We measured the results by **Success Rate (SR)**, **Min Epoch (ME)**, and **N**, as described in the caption of Table 2.7.

λ_c regularizes the shape of the NCBF by penalizing incorrectly categorized samples. Table 2.7a indicates that when λ_c is too small, the training procedure fails to train an NCBF that correctly separates the safe and unsafe regions, resulting in failure of certification. Meanwhile, a larger weight delivers similarly good performance.

λ_f penalizes violations of Lie derivative conditions. Table 2.7b shows that the result is not sensitive to this hyperparameter, as this term quickly goes down to 0 when the Lie derivative condition is satisfied. We note that over-penalizing this condition should be avoided since the NCBF would otherwise learn an unrecoverable incorrect shape, as demonstrated by the failure case when $\lambda_f = 8$.

$\lambda_{\mathcal{B}}$ the boundary regularization term reduces the number of boundary hyperplanes and benefits convergence.

Table 2.7c shows the importance of the term k used in the modified sigmoid function. Since this term appears in the exponential part of the sigmoid function, when it is too large, it leads to gradient explosions during backpropagation, which crashes the training process. Conversely, a reasonably larger k better approximates the activation pattern, leading to a reduced number of boundary hyperplanes.

In summary, balancing the hyperparameters is relatively straightforward, as the training performance remains robust across a wide range of hyperparameter values. When training failures do occur, we can systematically identify the cause from observation. This enables proper guidance in choosing and adjusting the appropriate hyperparameters.

2.8 Conclusion

This chapter studied the problem of synthesizing and verifying safety of a nonlinear control system using a NCBF represented by a feed-forward neural network with ReLU activation function. We leveraged a generalization of Nagumo's theorem for proving invariance of sets with nonsmooth boundaries to derive necessary and sufficient conditions for safety. The exact safety conditions addressed the issue that ReLU NCBFs will be nondifferentiable at certain points, thus invalidating traditional safety verification methods. Based on this condition, we proposed an algorithm for safety verification of NCBFs that first decomposes the NCBF into

piecewise linear segments and then solves a nonlinear program to verify safety of each segment as well as the intersections of the linear segments. We mitigated the complexity by only considering the boundary of the safe region and pruning the segments with IBP and LiRPA. To further mitigate the computational complexity, we proposed Synthesis with Efficient Exact Verification (SEEV), which co-designs the synthesis and verification components to enhance scalability of the verification process. We augment the NCBF synthesis with a regularizer that reduces the number of piecewise-linear segments at the boundary, and hence reduces the total workload of the verification. We then propose a verification approach that efficiently enumerates the linear segments at the boundary and exploits tractable sufficient conditions for safety. We evaluated our approach through numerical studies with comparison to state-of-the-art SMT-based methods.

Limitations: The method proposed in this chapter mitigated the scalability issue. However, the synthesis and verification of NCBFs for higher-dimensional systems is challenging. Exact verification of non-ReLU NCBFs, which lack ReLU's simple piecewise linearity, remains an open problem.

Chapter 3

Neural Control Barrier Functions For Stochastic Systems

In the previous chapter, we proposed NCBF synthesis and verification for deterministic systems. However, real systems must operate in the presence of stochastic noise, which could invalidate safety guarantees derived from deterministic NCBFs. Existing works on the verification of the NCBF focus on the synthesis and verification of NCBFs in deterministic settings, leaving the stochastic NCBFs (SNCBFs) less studied.

In this chapter, we propose a training framework to synthesize provably valid NCBFs for continuous-time, stochastic systems. Our methodology establishes completeness guarantees by deriving a validity condition, which ensures efficacy across the entire state space with only a finite number of data points. We consider the cases of smooth SNCBFs with twice-differentiable activation functions and SNCBFs that utilize ReLU activation functions. We also propose a verification-free synthesis framework for smooth SNCBFs and a verification-in-the-loop synthesis framework for both smooth and ReLU SNCBFs. We validate our frameworks in three cases, namely, the inverted pendulum, Darboux, and the unicycle model. We make the following contributions towards synthesizing and verifying NCBFs for stochastic systems.

- 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.
- We introduce the Stochastic NCBF (SNCBF) with ReLU activation functions and derive sufficient safety conditions using Tanaka's formula.

- We utilize the piecewise linearity of ReLU NNs, formulate the verification of ReLU SNCBFs as nonlinear programs and propose practical algorithms for efficient verification.
- We frame the VITL synthesis with efficient verifiers for both smooth and ReLU NCBFs synthesis. The VITL relaxes the dense sampling and single-layer assumption of the smooth SNCBF.
- We validate our approach in three cases, namely, the inverted pendulum, Darboux, and the unicycle model. The experiments illustrate that both smooth and ReLU SNCBFs can output verifiably safe results while covering a larger safe subset compared to the baseline approach of a Fault-Tolerant SNCBF without VITL.

The remainder of this chapter is organized as follows. Section 3.1 presents the dynamics model and preliminary results. Section 3.2 presents the verifiable synthesis, verification, and VITL synthesis of SNCBFs with smooth activation functions. Section 3.3 introduces ReLU SNCBFs with sufficient safety conditions, derives verification conditions, and VITL synthesis. Section 3.4 presents the validation of the proposed methods with experimental settings, results, and comparison in three case studies. Section 3.5 concludes the chapter.

3.1 Preliminaries

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

3.1.1 System Model

We consider a continuous time stochastic control system with state $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ 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):

$$dx_t = (f(x_t) + g(x_t)u_t)\ dt + V\ dv_t, \tag{3.1}$$

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, v_t is n_v-dimensional Brownian motion and $V \in \mathbb{R}^{n_x \times n_v}$. We make the following assumptions on system (3.1).

Assumption 3.1. *There exists an initial state $\hat{x} \in \mathcal{X}$ such that $\mathbb{P}[x_0 = \hat{x}] = 1$. The SDE* (3.1) *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

$$\mathcal{X}_S = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) \geq 0\} \tag{3.2}$$

$$\mathcal{X}_U = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) < 0\}. \tag{3.3}$$

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 next present the definitions of class-κ functions, extended class-κ functions, and indicator functions. A continuous function $\alpha : [0, d) \to [0, \infty)$ for some $d > 0$ is said to belong to *class-κ* if it is strictly increasing and $\alpha(0) = 0$. Here, d is allowed to be ∞. If function can be extended to the interval $\alpha : (-b, d) \to (-\infty, \infty)$ with $b > 0$ (which is also allowed to be ∞), we call it an *extended class-κ function* denoted as κ_∞. We denote the *indicator function* as $\mathbb{1}_{\mathcal{C}}(x)$ which takes the value of 1 if $x \in \mathcal{C}$ and 0 if $x \notin \mathcal{C}$. 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 $\{\omega - a\}_- := -\min\{\omega - a, 0\}$

Notations: We denote the stochastic process of the state x as x_t and let 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.

3.1.2 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 3.1 (Tanaka's Formula [103], 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,*

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

A detailed treatment of semimartingales and Tanaka's formula can be found in [103]. Consider a function $B : \mathbb{R}^{n_x} \to \mathbb{R}$ mapping from a state x to a scalar. The infinitesimal generator of B is defined as follows.

Definition 3.1 (Infinitesimal generator)**.** *Let x_t be the strong solution to* (3.1)*. The infinitesimal generator $\mathcal{A}$ of x_t is defined by*

$$
\mathcal{A}B(x) = \lim_{t \downarrow 0} \frac{\mathbb{E}\left[B\left(x_t\right) \mid x_0 = x\right] - B(x)}{t}; \quad x \in \mathbb{R}^{n_x}
$$

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

We make the following assumption on $B(x)$.

Assumption 3.2. *We assume the function $B(x)$ 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 $\mathbb{E}\left[B\left(x_t\right) \mid x_0 = x\right]$ is well-defined and finite.*
- *The limit defining $\mathcal{A}B(x)$ exists for every $x \in \mathcal{X}$.*

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

3.1.3 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 3.2 (Stochastic Zeroing CBF)**.** *A continuously differentiable function $B : \mathcal{X} \to \mathbb{R}$ is called a stochastic zeroing control barrier function for system* (3.1) *if the following conditions are satisfied.*

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

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

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

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

$$\begin{aligned} u^*(x,t) = \min_{u \in \mathcal{U}} \|u - u_{ref}(x,t)\|^2 \\ \text{s.t. } \mathcal{A}B(x) + \alpha(B(x)) \geq 0. \end{aligned} \tag{3.4}$$

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

Next, we present the worst-case probability estimation.

Proposition 3.1 ([104, Proposition III.5]). *Suppose the map $B(x)$ is a Stochastic CBF with a linear class-α function (where $\alpha(x) = kx; k > 0$), and the control strategy as $\mathcal{U}_z = \{u \in \mathcal{U} : \mathcal{A}B(x) + kB(x) \geq 0\}$. Let $c = \sup_{x \in \mathcal{C}} B(x)$ and $x_0 \in int(\mathcal{D})$, then under any $u \in \mathcal{U}_z$ we have the following worst-case probability estimation:*

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), 0 \leq t \leq T\right] \geq \left(\frac{B(x_0)}{c}\right) e^{-cT}.$$

Neural CBFs

In what follows, we briefly review the concept of Neural CBF from Chapter. 2. CBFs that are defined by neural networks, denoted as Neural Control Barrier Functions (NCBFs), have been proposed to leverage 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 σ.

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

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.

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

where σ is an elementary activation function. We denote $z_{ijt} = z_{ij}(x_t)$. The output of the network is given by $B(x) = W_L^T \mathbf{z}_L + r_L$. The j-th neuron at the i-th layer is *activated* by a particular input 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 [39, Lemma 1]

$$\mathcal{X}(\mathbf{S}) \triangleq \bigcap_{i=1}^{L} \left(\bigcap_{j \in \mathcal{S}_i} \{x : W_{ij}^T(\overline{\mathbf{W}}_{i-1}(\mathbf{S})x + \overline{\mathbf{r}}_{i-1}) + r_{1j} \geq 0\} \right.$$
$$\left. \cap \bigcap_{j \notin \mathcal{S}_i} \{x : W_{ij}^T(\overline{\mathbf{W}}_{i-1}(\mathbf{S})x + \overline{\mathbf{r}}_{i-1}) + r_{1j} \leq 0\} \right). \quad (3.5)$$

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 $\mathbf{T}' \subseteq \mathbf{T}(\mathbf{S}_1, \ldots, \mathbf{S}_r)$, $(\bigcap_{l=1}^{r} \mathcal{X}(\mathbf{S}_l)) \cup \mathbf{T}' \subseteq \{\mathcal{X}(\mathbf{S}_1), \ldots, \mathcal{X}(\mathbf{S}_r)\}$.

3.1.4 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 $p(x)$ is Sum-Of-Squares (SOS) if and only if it can be written as

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

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 $\phi_1, \ldots, \phi_{k_\Sigma}$ is 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

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

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 3.2 (Positivstellensatz [105])**.** *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*

$$\left\{ x \in \mathbb{R}^{n_x} \;\middle|\; \begin{array}{ll} \phi_i(x) \geq 0, & i = 1, \ldots, k_\Sigma \\ \chi_j(x) \neq 0, & j = 1, \ldots, k_\mathcal{M} \\ \Gamma_l(x) = 0, & l = 1, \ldots, k_\mathbb{I} \end{array} \right\} \tag{3.6}$$

   is empty.

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

3.2 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 3.1. *Given a continuous-time stochastic control system defined as* (3.1)*, a desired confidence level η, state space $\mathcal{X}$, initial safe set and the unsafe set $\mathcal{X}_I \subseteq \mathcal{X}_S$ and $\mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_S$, respectively, devise an algorithm that synthesizes a stochastic neural CBF* (SNCBF) *$B(x)$ with continuously differentiable (smooth) activation functions. In particular, the SNCBF must satisfy the following worst-case probability guarantee:*

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), 0 \leq t \leq T \mid x_0 \in int(\mathcal{D})\right] \geq 1 - \eta.$$

Our strategy to address Problem 3.1 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 3.1, 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) $B(x)$ with twice-differentiable activation functions is given by [106] as:

$$\mathcal{A}B = \frac{\partial B}{\partial x}\left(f(x) + g(x)u\right) + \frac{1}{2}\mathsf{tr}\left(V^{\intercal}\frac{\partial^2 B}{\partial x^2}V\right). \tag{3.7}$$

3.2.1 Smooth SNCBF Verifiable Synthesis

The conditions of Problem 3.1 are satisfied if we can synthesize a NN $B(x)$ 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 x_t to a control input $u_t = \mu(x_t)$ at each time t. $B(x)$ ensures safety by imposing conditions on system states and control inputs defined as follows.

$$\begin{aligned}
&B(x) \geq 0, \forall x \in \mathcal{X}_I, \\
&B(x) < 0, \forall x \in \mathcal{X}_U, \\
&\frac{\partial B(x)}{\partial x}(f(x) + g(x)u) + \frac{1}{2}\mathsf{tr}\left(V^\intercal \frac{\partial^2 B}{\partial x^2} V\right) + \alpha\left(B(x)\right) \geq 0, \forall x \in \mathcal{D}.
\end{aligned} \tag{3.8}$$

To approach Problem 3.1, we make the following assumptions in Section 3.2.1.

Assumption 3.3. *For the smooth SNCBF verifiable synthesis, we assume that*

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

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

$$\text{ROP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & \max\left(q_k(x)\right) \leq \psi, k \in \{1,2,3\} \\ & \forall x \in \mathcal{X}, \psi \in \mathbb{R}, \end{cases} \tag{3.9}$$

where

$$\begin{aligned} q_1(x) &= (-B(x))\,\mathbb{1}_{\mathcal{X}_I}(x), \\ q_2(x) &= (B(x)+\delta)\,\mathbb{1}_{\mathcal{X}_U}(x), \\ q_3(x) &= -\frac{\partial B}{\partial x}\left(f(x)+g(x)u\right) - \frac{1}{2}\mathsf{tr}\left(V^\intercal \frac{\partial^2 B}{\partial x^2} V\right) - \alpha(B(x)), \end{aligned} \tag{3.10}$$

where δ is a small positive scalar ensuring strict inequality. The ROP in (3.9) 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 $x_i \in \mathcal{X}, i \in \{1,\ldots,N\}$. Let $\bar{\epsilon}$ be a positive scalar. We sample points dense enough such that $\|x - x_i\| \leq \bar{\epsilon}$ for any $x \in \mathcal{D}$ and some x_i. We then solve:

$$\text{SOP}: \begin{cases} \min\limits_{\psi} & \psi \\ \text{s.t.} & q_1(x_i) \leq \psi, \forall x_i \in \mathcal{S}, \\ & q_2(x_i) \leq \psi, \forall x_i \in \mathcal{U}, \\ & q_3(x_i) \leq \psi, \forall x_i \in \mathcal{D}, \\ & \psi \in \mathbb{R} \end{cases} \tag{3.11}$$

where $q_k(x), k \in 1,2,3$ are as defined in (3.10), 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 $B(x)$ obtained via (3.11) provides a valid solution to Problem 3.1.

Theorem 3.3. *Consider a continuous time stochastic control system* (3.1)*, and initial safe and unsafe sets $\mathcal{X}_I \subseteq \mathcal{X}_S \subseteq \mathcal{X}$ and $\mathcal{X}_U \subseteq \mathcal{X} \setminus \mathcal{X}_S$, respectively. Let $B(x)$ be the SNCBF with trainable parameters θ. Suppose condition 1) in Assumption 3.3 holds so that the functions $q_k(x), k \in \{1,2,3\}$ in equation* (3.10) *are Lipschitz continuous. For the SOP* (3.11) *constructed by utilizing $x_1,\ldots,x_N$ such that $\mathcal{D}$ is covered by balls centered at the x_i's with radii $\bar{\epsilon}$, let ψ^* be the optimal value. Then $B(x)$ is a valid* SNCBF*, i.e., it solves Problem 3.1, if the following condition holds:*

$$L_{\max}\bar{\epsilon} + \psi^* \leq 0, \tag{3.12}$$

where $L_{\max}$ is the maximum of the Lipschitz constants of $q_k(x), k \in \{1,2,3\}$ in (3.10).

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 (3.12), then the $B(x)$ is a valid SCBF, satisfying conditions (3.8). □

Theorem 3.3 reformulates Problem 3.1 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 (3.11) and ψ^* satisfies condition (3.12).

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 [107, 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

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

is positive semidefinite.

Our scenario necessitates ensuring not only the Lipschitz boundedness of the SNCBF $B(x)$, but also of $\frac{\partial B}{\partial x}$ and $V^\intercal \frac{\partial^2 B}{\partial x^2} V$. Therefore, we must explore the relationship 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 $B(x)$, its derivative, and the Hessian terms in (3.10).

Theorem 3.4. *Suppose Assumption 3.3 holds for an SNCBF $B(x)$. The certificate for L-Lipschitz continuity of $\frac{\partial B}{\partial x}$ which is the derivative of the NN is given by $M_{\hat{\sigma}}(\hat{\theta}, \Omega) \succeq 0$, where $\hat{\sigma} = \sigma'$ and $\hat{\theta} = (\mathbf{W}_1, \hat{\mathbf{W}}_2)$, $\hat{\mathbf{W}}_2$ is defined as:*

$$\hat{\mathbf{W}}_2 = \mathbf{W}_1^T \mathsf{diag}(\mathbf{W}_2). \tag{3.13}$$

Additionally, the certificate for the L-Lipschitz continuity of $\mathsf{tr}(V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V)$ is expressed as $M_{\bar{\sigma}}(\bar{\theta}, \Omega) \succeq 0$, where $\bar{\sigma} = \sigma''$ and $\bar{\theta} = (\mathbf{W}_1, \bar{\mathbf{W}}_2)$ and $\bar{\mathbf{W}}_2$ is defined as:

$$\bar{\mathbf{W}}_2 = \begin{bmatrix} \sum_{j=0}^{r} V_j^2 \mathbf{W}_2^{j1} \mathbf{W}_1^{1j} & \dots & \sum_{j=0}^{r} V_j^2 \mathbf{W}_2^{jn} \mathbf{W}_1^{pj} \end{bmatrix}. \tag{3.14}$$

Proof. Consider an NN with p neurons in a hidden layer. The dimension of the input (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:

$$\begin{aligned} \mathbf{y} &= \mathbf{W}_2 \sigma(\mathbf{W}_1 x + \mathbf{r}_1) \\ \frac{\partial \mathbf{y}}{\partial x} &= \mathbf{W}_2 \mathsf{diag}(\sigma') \mathbf{W}_1 \\ &\quad (\text{Here, } \sigma' = \sigma'(\mathbf{W}_1 x + \mathbf{r}_1)). \end{aligned}$$

The dimension of $\frac{\partial \mathbf{y}}{\partial x}$ is $1 \times n_x$, therefore, its transpose will have the dimension of $n_x \times 1$.

$$\begin{aligned} (\frac{\partial \mathbf{y}}{\partial x})^T &= (\mathbf{W}_2 \mathsf{diag}(\sigma') \mathbf{W}_1)^T \\ &= ((\sigma')^T \mathsf{diag}(\mathbf{W}_2) \mathbf{W}_1)^T \\ &= \underbrace{\mathbf{W}_1^T \mathsf{diag}(\mathbf{W}_2)}_{\hat{\theta}_l} \sigma'(\mathbf{W}_1 x + \mathbf{r}_1). \end{aligned}$$

The term derivative term $(\frac{\partial \mathbf{y}}{\partial x})$ is equivalent to an NN with with activation $\hat{\sigma} = \sigma_l'$ and weight parameters $\hat{\theta} = (\mathbf{W}_1, \hat{\mathbf{W}}_2)$ and $\hat{\mathbf{W}}_2$ is defined as:

$$\hat{\mathbf{W}}_2 = \mathbf{W}_1^T \mathsf{diag}(\mathbf{W}_2).$$

Therefore, the certificate for L-Lipschitz continuity of the derivative term $(\frac{\partial \mathbf{y}}{\partial x})$ is given by $M_{\hat{\sigma}}(\hat{\theta}, \Omega) \succeq 0$.

Now, let us calculate $\mathsf{tr}(V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V)$, where V is $n_x \times n_x$ diagonal matrix.

$$\begin{aligned}
\frac{\partial \mathbf{y}}{\partial x} &= \hat{\mathbf{W}}_2 \hat{\sigma}(\mathbf{W}_1 x + \mathbf{r}_1) \\
V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V &= V^T (\hat{\mathbf{W}}_2 \mathsf{diag}(\hat{\sigma}') \mathbf{W}_1) V \\
\mathsf{tr}(V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V) &= \mathsf{tr}(V^T (\hat{\mathbf{W}}_2 \mathsf{diag}(\hat{\sigma}') \mathbf{W}_1) V)
\end{aligned}$$

$$= \underbrace{\left[\textstyle\sum_{j=0}^{r} V_j^2 \hat{\mathbf{W}}_2^{j1} \mathbf{W}_1^{1j} \dots \sum_{j=0}^{r} V_j^2 \hat{\mathbf{W}}_2^{jp} \mathbf{W}_1^{pj} \right]}_{\bar{\mathbf{W}}_2} \sigma''(\mathbf{W}_1 x + \mathbf{r}_1).$$

The term $\mathsf{tr}(V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V)$ is equivalent to a feedforward NN with activation $\bar{\sigma} = \sigma''$ and weight parameters $\bar{\theta} = (\mathbf{W}_1, \bar{\mathbf{W}}_2)$ and $\hat{\mathbf{W}}_2$ is defined as:

$$\bar{\mathbf{W}}_2 = \begin{bmatrix} \sum_{j=0}^{n_x} V_j^2 \hat{\mathbf{W}}_2^{j1} \mathbf{W}_1^{1j} & \dots & \sum_{j=0}^{n_x} V_j^2 \hat{\mathbf{W}}_2^{jp} \mathbf{W}_1^{pj} \end{bmatrix}.$$

Therefore, the certificate for Lipschitz continuity of the $\mathsf{tr}(V^T \frac{\partial^2 \mathbf{y}}{\partial x^2} V)$ term is given by $M_{\bar{\sigma}}(\bar{\theta}, \Omega) \succeq 0$. □

Next, we define a suitable loss function to train the smooth SNCBF $B(x)$ such that its minimization yields a solution to Problem 3.1:

$$\begin{aligned}
\mathcal{L}(\theta) =& \frac{1}{N} \sum_{x_i \in \mathcal{S}} \max\left(0, (-B(x_i))\, \mathbb{1}_{\mathcal{X}_I} - \psi\right), \\
&+ \frac{1}{N} \sum_{x_i \in \mathcal{U}} \max\left(0, (B(x_i) + \delta)\, \mathbb{1}_{\mathcal{X}_U} - \psi\right), \\
&+ \frac{1}{N} \sum_{x_i \in \mathcal{D}} \max(0, -\frac{\partial B}{\partial x} f(x_i) + \frac{\partial B}{\partial x} g(x_i) u + \\
&\frac{1}{2} \mathsf{tr}\left(V^\intercal \frac{\partial^2 B(x_i)}{\partial x^2} V \right) - \alpha(B(x_i)) - \psi).
\end{aligned} \tag{3.15}$$

Let us consider a constrained optimization problem aiming to minimize loss $\mathcal{L}(\theta)$ in (3.15) subject to $M_j(\theta, \Omega) \succeq 0$ for $j = 1, 2, 3$, to ensure the Lipschitz continuity of the SNCBF $B(x)$,

as well as its derivative and Hessian terms. By employing a log-determinant barrier function, we convert this into an unconstrained optimization problem:

$$\min_{\theta,\Omega} \mathcal{L}\left(f_\theta\right) + \mathcal{L}_M(\theta,\Omega),$$

where $\mathcal{L}_M(\theta,\Omega) = -\sum_{j=0}^{q} \rho_j \log\det\left(M_j(\theta,\Omega)\right)$ 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

$$\begin{aligned}\mathcal{L}_M(\theta,\Omega,\hat{\Omega},\bar{\Omega}) = &-c_{l_1}\log\det(M_1(\theta,\Omega)) \\ &- c_{l_2}\log\det(M_2(\hat{\theta},\hat{\Omega})) - c_{l_3}\log\det(M_3(\bar{\theta},\bar{\Omega})),\end{aligned} \tag{3.16}$$

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 3.4.

Finally, let us consider the following loss function to satisfy validity condition (3.12):

$$\mathcal{L}_v(\psi) = \max\left(0, L_{\max}\bar{\epsilon} + \psi\right), \tag{3.17}$$

where $L_{\max}$ is maximum of the Lipschitz constants of $q_k(x), k \in \{1,2,3\}$ in (3.10), 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 $\bar{\epsilon}$, $\mathcal{L}_h$, $\mathcal{L}_{dh}$, $\mathcal{L}_{d2h}$, ω_1, ω_2, c_{l_1}, c_{l_2}, c_{l_3}, and the maximum number of epochs. The overall algorithm is summarized in Algorithm 5.

3.2.2 Smooth SNCBF Verification and Synthesis

The previous approach is limited to single hidden layer NNs and requires dense sampling under Assumption 3.3. In this subsection, we consider an NN with multiple hidden layers and relax assumptions on Lipschitz continuity and diagonal V. To ensure the conditions in (3.8) hold, we propose a novel verification to validate a smooth SNCBF. We then leverage

Algorithm 5 Training Formally Verified SNCBF

Require: Data Sets: $\mathcal{S}, \mathcal{U}, \mathcal{D}$, Dynamics: f, g, σ, Lipschitz Bounds: $L_h, L_{dh}, L_x, L_{d2h}$
  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**
    $B \leftarrow \theta$
    $u_i \leftarrow \text{SNCBF\_QP}(B, f, g, \sigma, x_i, \psi)$ ▷ From eq. (3.4)
    $\mathcal{L}_\theta \leftarrow (B, f, g, \sigma, x_i, u_i, \psi)$ ▷ From eq. (3.15)
    $\theta \leftarrow Learn(\mathcal{L}_\theta, \theta)$
    $\mathcal{L}_M \leftarrow (\theta, \Omega, \Omega', \Omega'')$ ▷ From eq. (3.16)
    $\theta, \Omega, \Omega', \Omega'' \leftarrow Learn(\mathcal{L}_M)$
    $\mathcal{L}_v \leftarrow (L_{\max}, \psi)$ ▷ From eq. (3.17)
    $\psi \leftarrow Learn(\mathcal{L}_v)$
  end while

the Verification-In-The-Loop (VITL) synthesis proposed in [16], propose SMT-based verifiers for smooth SNCBFs, and construct the loss function for the VITL synthesis.

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 3.3 (Valid Smooth SNCBFs)**.** *The function $B(x)$ is called a valid smooth SNCBF if $B(x)$ satisfies the* Correctness *requirement of SNCBFs, i.e. $\forall x \in \mathcal{D}$, and the* Feasibility *requirement of SNCBFs i.e. $x \in \mathcal{X}_S$, and there exists a stochastic smooth NCBF control u satisfying the conditions of Proposition 3.1.*

These correctness and feasibility conditions must hold for every state in the super-0-level-set $x \in \mathcal{D}$. 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 [34, 39], it is essential to verify SNCBF.

As shown in Fig. 3.1, 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 $B(x)$ that satisfies the conditions of Definition 3.3 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

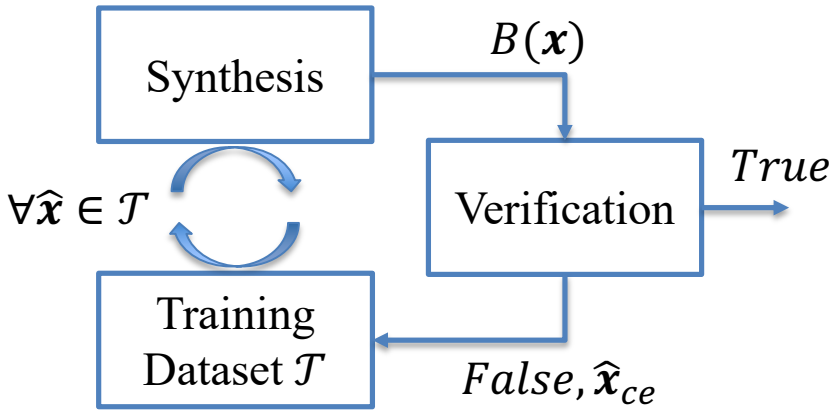


Figure 3.1: Workflow of the synthesis with verification in the loop.

weighted sum of two components, structured as an unconstrained optimization problem to seek out θ.

$$\min_{\theta} \quad \lambda_f \mathcal{L}_f(\mathcal{T}) + \lambda_c \mathcal{L}_c(\mathcal{T}) \tag{3.18}$$

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 (3.22a)-(3.22c) (feasibility condition of Definition 3.3), and the term $\mathcal{L}_c(\mathcal{T})$ penalizes the correctness counterexample with negative minima of (3.20) (correctness condition of Definition 3.3). For each sample $\hat{x} \in \mathcal{T}$ the safe control signal $u(x)$ 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 $\mathcal{L}_f = ||u(x) - u_{ref}(x)||^2 + r$. The loss term $\mathcal{L}_c$ is defined as

$$\mathcal{L}_c = \frac{1}{N} \sum_{\hat{x}_i \in \mathcal{X}_I} \max\left(0, (\epsilon - B(\hat{x}_i))\, \mathbb{1}_{\mathcal{X}_s}\right) + \frac{1}{N} \sum_{\hat{x}_j \in \mathcal{X}_U} \max\left(0, (B(\hat{x}_j) + \epsilon)\, \mathbb{1}_{\mathcal{X}_U}\right) \tag{3.19}$$

The outer loop validates a given SNCBF $B(x)$ by searching for counterexamples $\hat{x}_{ce}$ and updates the training dataset as $\mathcal{T} \cup \{\hat{x}_{ce}\}$. The intuition behind the framework is to penalize violations for all $\hat{x} \in \mathcal{T}$ and generalize it from $\mathcal{T}$ to $\mathcal{X}$ through the verification module. 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 3.3. 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 $x \in \mathcal{D} \subseteq \mathcal{X}_S$ with $h(x) < 0$. Specifically, we check if $\mathcal{D} \cap \mathcal{X}_U = \emptyset$. It suffices to solve the nonlinear program

$$\begin{aligned} \min_x \quad & h(x) \\ \text{s.t.} \quad & B(x) \geq 0. \end{aligned} \tag{3.20}$$

If there exists x^* such that $h(x^*) < 0$ while $B(x^*) = 0$, then x^* is a correctness counterexample.

Next, we verify the feasibility condition in Definition 3.3. Let $\lambda(x_t)$ and $\xi(x_t)$ be as follows.

$$\lambda(x_t) := \frac{\partial B}{\partial x_t} g(x_t)$$

$$\xi(x_t) := \frac{\partial B}{\partial x_t} f(x_t) + \frac{1}{2}\mathbf{tr}\left(V^T \frac{\partial^2 B}{\partial x_t^2} V\right) - \alpha(B(x_t)).$$

Rearranging the terms in (3.8), we have the affine constraints given as

$$\lambda(x_t) u_t \leq \xi(x_t), \tag{3.21}$$

The feasibility counterexample refers to the state x such that there does not exist a u satisfying condition (3.21).

We next consider the case with control input constraints $u \in \mathcal{U} := \{u : Au \leq \mathbf{b}\}$. We present $\Lambda(x_t)$, $\Xi(x_t)$ and corresponding verification formulation.

Proposition 3.2. *Let $\Lambda(x_t)$ and $\Xi(x_t)$ be as follows.*

$$\Lambda(x_t) := \begin{bmatrix} -\lambda(x_t) \\ A \end{bmatrix}; \qquad \Xi(x_t) := \begin{bmatrix} \xi(x_t) \\ \mathbf{b} \end{bmatrix}$$

There always exists a u satisfying (3.21) if and only if there is no $x \in \mathcal{D}$ and $\mathbf{y} \in \mathbb{R}^{n_u+1}$ such that

$$[\mathbf{y}]_i \geq 0, \ \forall i \in \{1, \ldots, n_u + 1\} \tag{3.22a}$$

$$-\mathbf{y}^T \Xi(x) < 0 \tag{3.22b}$$

$$\left[\mathbf{y}^T \Lambda(x)\right]_i = 0, \ \forall i \in \{1, \ldots, n_u + 1\} \tag{3.22c}$$

Proof. The proof is to show that there does not exist x such that, for any $u \in \mathcal{U}$, (3.21) fails to hold. By Farkas's Lemma, the existence of control input u satisfying (3.21) is equivalent to the non-existence of $\mathbf{y}$ satisfying (3.22a)–(3.22c), completing the proof. □

The NN is a polynomial if all the activation functions are polynomials, e.g., Hermite Polynomial Activation Functions [108]. The following lemma shows that given a polynomial SNCBF, the non-existence of x and $\mathbf{y}$ can be proved through Positivstellensatz.

Lemma 3.1. *There is no feasibility counterexample $x \in \mathcal{D}$ if and only if there exist polynomials $\rho_1^{\mathbf{y}}(x) \dots \rho_{n_u+1}^{\mathbf{y}}(x)$, sum-of-squares polynomials $q_P(x)$, integers $\tau = n_u + 3$, r_1 such that*

$$\phi(x, \mathbf{y}) + \chi(x, \mathbf{y}) + \Gamma(x, \mathbf{y}) = 0, \tag{3.23}$$

and

$$\begin{aligned}
\phi(x, \mathbf{y}) &= \sum_{K \subseteq \{1,\dots,\tau\}} p_i(x) \prod_{i \in K} \phi_i(x, \mathbf{y}) \\
\chi(x, \mathbf{y}) &= \phi_1(x, \mathbf{y})^{2r_1} \\
\Gamma(x, \mathbf{y}) &= \sum_{i=1}^{n_u} \rho_i^{\mathbf{y}} \left[\mathbf{y}^T \Lambda(x)\right]_i,
\end{aligned}$$

where $\phi_1(x) = -\mathbf{y}^T \Xi(x)$, $\phi_2(x) = B(x)$, and $\phi_{3,\dots,n_u+3}(x, \mathbf{y}) = y_i$.

Proof. By Proposition 3.2, we have that there is no feasibility counterexample iff there exists no $x \in \mathcal{D}$ and $\mathbf{y}$ such that (3.22a)–(3.22c) hold. The condition $\mathbf{y}^T \Xi(x) < 0$ is equivalent to $-\mathbf{y}^T \Xi(x) \geq 0$ and $\mathbf{y}^T \Xi(x) \neq 0$. The result then follows from the Positivstellensatz □

If the SNCBF has non-polynomial activation functions, the conditions of Proposition 3.2 can be verified by solving the nonlinear program

$$\begin{aligned}
\min_{x,y} \quad & \mathbf{y}^T \Xi(x) \\
\text{s.t.} \quad & B(x) \geq 0 \\
& [\mathbf{y}]_i \geq 0, \ \forall i \in \{1, \dots, n_u + 1\} \\
& \left[\mathbf{y}^T \Lambda(x)\right]_i = 0, \ \forall i \in \{1, \dots, n_u + 1\}
\end{aligned} \tag{3.24}$$

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 3.1. *There exists $u \in \mathbb{R}^{n_u}$ satisfying*

$$-\lambda(x_t)u \leq \xi(x_t)$$

for all $x \in \mathcal{D}$ if and only if there is no x such that $\lambda(x) = \mathbf{0}$, and $\xi(x) > 0$.

Proof. When the vector $\frac{\partial B}{\partial x} g(x) \neq \mathbf{0}$, there is always $u \in \mathbb{R}^{n_u}$ that satisfies (3.21). When the vector $\frac{\partial B}{\partial x} g(x) = \mathbf{0}$, u does not affect the inequality, which is equivalent to having $u = \mathbf{0}$. In this case, $\Xi(x_t) > 0$ ensures (3.21) hold. □

Under the condition of Corollary 3.1, the nonlinear program (3.24) reduces to

$$\begin{aligned} \min_x \quad & \xi(x) \\ \text{s.t.} \quad & B(x) \geq 0 \\ & [\lambda(x)]_i = 0, \ \forall i \in \{1, \ldots, n_u\} \end{aligned} \tag{3.25}$$

The condition of Corollary 3.1 can be verified by checking whether the optimal value is nonnegative (safe) or negative (unsafe). Nonlinear programs (3.20), (3.24), and (3.25) can be solved by SMT solvers with numerical completeness guarantees.

3.3 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.

Problem 3.2. *Given a continuous-time stochastic control system defined as* (3.1)*, 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) $B_\theta(x)$ *with ReLU activation functions. In particular, the SNCBF must satisfy the following worst-case probability guarantee:*

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), 0 \leq t \leq T \mid x_0 \in int(\mathcal{D})\right] \geq 1 - \eta.$$

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.

3.3.1 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 $z_{1jt} = W_{1j}^T x_t + r_{1j}$ and let $\mathcal{S}(t) = \{j : W_{1j}^T x_t + r_{1j} > 0\}$, i.e., the set of neurons whose pre-activation inputs are positive at time t.

By Tanaka's formula, we have the random process $B(x_t)$ satisfies

$$B(x_t) = \sum_{j=1}^{M} \left[W_{2j}(\{z_{1j0}\}_+ + \int_0^t \mathbb{1}(z_{1js} > 0) dz_{1js})\right] + \frac{1}{2}\sum_{j=1}^{M} W_{2j}(|z_{1jt}| - |z_{1j0}| - \int_0^t \text{sgn}(z_{1js}) dz_{1js}) + r_2. \quad (3.26)$$

Let $\mathcal{D} := \{x : B(x) \geq 0\}$. We next derive a stochastic process that lower bounds $B(x_t)$.

Lemma 3.2. *Suppose that there exists a scalar R_j such that $|z_j| \leq R_j$ for all $j \in \{1, \ldots, M\}$ whenever $B(x_t) \geq 0$. Let $\tilde{B}(x_t)$ be defined as follows.*

$$\tilde{B}(x_t) := \sum_{j=1}^{M} [W_{2j}(\{z_{1j0}\}_+ + \int_0^t \mathbb{1}(z_{1js} > 0)\ dz_{1js})] + \frac{1}{2}\sum_{j=1}^{M} W_{2j}(-|z_{1j0}| - \int_0^t sgn(z_{1js})\ dz_{1js}) - \frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2 + r_2 \quad (3.27)$$

Then we have $\tilde{B}(x_t) \leq B(x_t)$ for all $x_t \in \mathcal{D}$.

Proof. Given that there exists a scalar R_j such that $|z_{1jt}| \leq R_j$ for all $j \in \{1, \ldots, M\}$ whenever $B(x_t) \geq 0$, 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 $\tilde{B}(x_t) \leq B(x_t)$ for all $x_t \in \mathcal{D}$. □

For simplicity, we let $\zeta(x_s) = 2x_s^T W_{1j} W_{1j}^T + 2r_{1j} W_{1j}^T$. The expected instantaneous rate of change of $\tilde{B}_t := \tilde{B}(x_t)$ is described by the Infinitesimal Generator $\mathcal{A}\tilde{B}_t$ defined as follows.

Lemma 3.3. *The infinitesimal generator of $\tilde{B}_t$ is*

$$\mathcal{A}\tilde{B}_t = \beta_D(x_t, u_t) \quad (3.28)$$

where

$$\begin{aligned} \beta_D(\cdot) \;=\; & \frac{1}{2}\sum_{j=1}^{M} W_{2j}\left(\mathbb{1}(z_{1js} > 0) - \frac{1}{2} sgn(z_{1js})\right) \\ & \cdot W_{1j}^T(f(x_s) + g(x_s)u_s) \\ & -\frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j}\Big[\zeta(x_s)(f(x_s) + g(x_s)u_s) \\ & +\frac{1}{2}\mathbf{tr}(V_s^T W_{1j} W_{1j}^T V_s)\Big]. \end{aligned}$$

Proof. Consider $\tilde{B}(x_t)$ defined by

$$\tilde{B}(x_t) = \tilde{B}_0 - \frac{1}{2}\sum_{j=1}^{M}\frac{|W_{2j}|}{R_j}z_{1jt}^2 + \sum_{j=1}^{M}W_{2j}\int_0^t(\mathbb{1}(z_{1js}>0) - \frac{1}{2}\text{sgn}(z_{1js}))dz_{1js},$$

where $\tilde{B}_0 = \sum_{j=1}^{M}(W_{2j}(\{z_{1j0}\}_+ - \frac{1}{2}|z_{1j0}|) + r_2$. By Ito's lemma [109], we have

$$dz_{1jt} = W_{1j}^T dx_t = W_{1j}^T(f(x_t) + g(x_t)u_t dt + W_{1j}^T V_t dv_t.$$

$$z_{1jt}^2 = z_{1j0}^2 + \int_0^t\left[\zeta(x_s)(f(x_s)+g(x_s)u_s) + \frac{1}{2}\mathbf{tr}(V_s^T W_{1j}W_{1j}^T V_s)\right]ds + \int_0^t\zeta(x_s)V_s\,dv_s.$$

Substituting into the formula for $\tilde{B}(x_t)$ gives

$$\begin{aligned}\tilde{B}(x_t) &= \tilde{B}_0 + \sum_{j=1}^{M}W_{2j}\left[\int_0^t\left(\mathbb{1}(z_{1js}>0) - \frac{1}{2}\text{sgn}(z_{1js})\right)\right.\\ &\quad\cdot W_{1j}^T(f(x_s)+g(x_s)u_s)\,ds\\ &\quad\left.+\int_0^t(\mathbb{1}(z_{1js}>0) - \frac{1}{2}\text{sgn}(z_{1js}))W_{1j}^T V_s\,dv_s\right]\\ &\quad-\frac{1}{2}\sum_{j=1}^{M}\frac{|W_{2j}|}{R_j}\left[\int_0^t(\zeta(x_s)(f(x_s)+g(x_s)u_s)\right.\\ &\quad\left.+\frac{1}{2}\mathbf{tr}(V_s^T W_{1j}W_{1j}^T V_s))ds + \int_0^t\zeta(x_s)V_s\,dv_s\right].\end{aligned}$$

This gives the drift term

$$\begin{aligned}\beta_D(x_s,u_s) &= \sum_{j=1}^{M}W_{2j}\left(\mathbb{1}(z_{1js}>0) - \frac{1}{2}\text{sgn}(z_{1js})\right)\\ &\quad\cdot W_{1j}^T(f(x_s)+g(x_s)u_s)\\ &\quad-\frac{1}{2}\sum_{j=1}^{M}\frac{|W_{2j}|}{R_j}\left[\zeta(x_s)(f(x_s)+g(x_s)u_s)\right.\\ &\quad\left.+\frac{1}{2}\mathbf{tr}(V_s^T W_{1j}W_{1j}^T V_s)\right].\end{aligned}$$

□

By (3.26) and (3.27), we have

$$\tilde{B}(x_t) = B(x_t) - \frac{1}{2}\sum_{j=1}^{M} W_{2j}|z_{1jt}| - \frac{1}{2}\sum_{j=1}^{M} \frac{|W_{2j}|}{R_j} z_{1jt}^2.$$

We next define the safe control constraint based on $\tilde{B}(x_t)$ as follows.

Definition 3.4. *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 $j \in \{1,\dots,M\}$, such that the following conditions are satisfied at each time t:*

$$\mathcal{A}\tilde{B}(x_t) \geq -k\tilde{B}(x_t). \tag{3.29}$$

Let $\tilde{\mathcal{D}} := \{x : \tilde{B}(x) \geq 0\}$. The following result shows the worst-case probability guarantee when applying stochastic ReLU NCBF control.

Theorem 3.5. *Suppose Assumption 3.1 holds and Assumption 3.2 holds for $\tilde{B}(x)$, i.e., $\tilde{B}(x)$ 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 $j \in \{1,\dots,M\}$ whenever $B(x) \geq 0$. Let $\mathcal{U}_z$ be a ReLU SNCBF control set in Definition 3.4, $x_0 \in \tilde{\mathcal{D}} \cap \mathcal{D}$ and $c = \sup_{x\in\tilde{\mathcal{D}}} \tilde{B}(x)$. By choosing $u_t \in \mathcal{U}_z$ for all time t, we have the following worst-case probability estimation:*

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), t \in [0,T]\right] \geq \left(\frac{\tilde{B}(x_0)}{c}\right) e^{-cT}.$$

Proof. Given $x_0 \in \tilde{\mathcal{D}} \cap \mathcal{D}$ and $u \in \mathcal{U}_z$, by Proposition 3.1, we have $x_t \in int(\tilde{\mathcal{D}})$ with probability

$$\mathbb{P}\left[x_t \in int(\tilde{\mathcal{D}}), t \in [0,T]\right] \geq \left(\frac{\tilde{B}(x_0)}{c}\right) e^{-cT}.$$

By Lemma 3.2, we have $\tilde{B}(x_t) \leq B(x_t)$ for all $x_t \in \mathcal{D}$.

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), t \in [0,T]\right] \geq \mathbb{P}\left[x_t \in int(\tilde{\mathcal{D}}), t \in [0,T]\right]$$

Since $\mathcal{D} \subseteq \mathcal{X}_S$, we have

$$\mathbb{P}\left[x_t \in int(\mathcal{D}), t \in [0,T]\right] \geq \mathbb{P}\left[x_t \in int(\mathcal{D}), t \in [0,T]\right].$$

□

3.3.2 ReLU SNCBF Verification and Synthesis

The safety guarantee derived in Section 3.3.1 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 3.5 (Valid ReLU SNCBFs)**.** *The function $B(x)$ is a valid ReLU SNCBF if $\forall x \in \mathcal{D}$ (*Correctness*) $x \in \mathcal{X}_S$ and (*Feasibility*) there exists control u satisfying the ReLU SNCBF $\tilde{B}(x)$ conditions* (3.29)*.*

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 $\mathcal{S}(x)$ denote the activated set $\mathcal{S}$ determined by 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 x_0, the following linear program checks the existence of a state x such that $B(x) \geq 0$.

$$\text{SuperLP}(\mathcal{S}(x_0)) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\{\mathcal{S}(x_0)\})^T x + \overline{r}(\{\mathcal{S}(x_0)\}) \geq 0 \\ & x \in \mathcal{X}(\mathcal{S}(x_0)). \end{cases}$$

The Feasibility of SuperLP means the hyperplane $\mathcal{X}(\mathcal{S}(x_0))$ contains the super-0-level-set of $B(x)$.

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 x such that $B(x) \geq 0$. 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 $x \in \mathcal{X}(\mathcal{S}) \cap \{x : W_{ij}x + r_{ij} = 0\}$

that satisfies $\overline{W}(\mathcal{S})x + \overline{r}(\mathcal{S}) \geq 0$. The unstable neuron linear program is defined as follows.

$$\text{USLP}(\mathcal{S}, j) = \begin{cases} \text{find} & x \\ \text{s.t.} & \overline{W}(\{\mathcal{S}\})^T x + \overline{r}(\{\mathcal{S}\}) \geq 0 \\ & W_{ij}(\{\mathcal{S}\})^T x + r_{ij}(\{\mathcal{S}\}) = 0 \\ & x \in \mathcal{X}(\mathcal{S}). \end{cases}$$

The enumeration process described above is summarized in Algorithm 6.

Algorithm 6 Enumerate Activated Sets

Input: $x_0 \in int(\mathcal{D})$ and $\mathcal{S}_0 = \mathcal{S}(\S_l)$
Output: Set of activation sets $\mathbf{Q}$
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 $j \in \{1, \ldots, M\}$ **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

Proposition 3.3. *Let* $\mathbf{Q}$ *denote the output of Algorithm 6. Then the super-0-level set* $\mathcal{D}$ *satisfies* $\mathcal{D} \subseteq \bigcup_{\mathcal{S} \in \mathbf{Q}} \mathcal{X}(\{\mathbf{S}\})$.

Proof. Suppose there exists $x' \in \mathcal{D} \setminus \left(\bigcup_{\mathcal{S} \in \mathbf{Q}} \mathcal{X}(\{\mathbf{S}\}) \right)$. Since $\mathcal{D}$ is connected, there exists a path γ such that $x' \in \gamma$. Let $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K$ 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 [110]. By the completeness, we have for any $\mathbf{T}' \subseteq \mathbf{T}(\mathcal{S}_0, \ldots, \mathcal{S}_K)$, $\left(\bigcap_{l=0}^{K} \mathcal{X}(\{\mathcal{S}_l\})\right) \cup \mathbf{T}' \subseteq \{\mathcal{X}(\{\mathcal{S}_1\}), \ldots, \mathcal{X}(\{\mathcal{S}_r\})\}$. Since $x' \in \mathcal{D} \setminus \left(\bigcup_{\mathcal{S}\in\mathbf{Q}} \mathcal{X}(\{\mathbf{S}\})\right)$, we have $x' = \gamma(t')$ and $x^* = \gamma(t)$ for some $t \in [0, t']$ such that $x^* \in \mathbf{T}'$. Then, we have the hinge $\mathbf{T}' \nsubseteq \mathbf{T}(\mathcal{S}_0, \ldots, \mathcal{S}_K)$, and $\mathbf{T}' \nsubseteq \{\mathcal{X}(\{\mathcal{S}_1\}), \ldots, \mathcal{X}(\{\mathcal{S}_r\})\}$, 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 3.5. We denote the state $x \in \mathcal{D} \subseteq \mathcal{X}_s$ 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

$$\begin{array}{ll} \min_x & h(x) \\ \text{s.t.} & \overline{W}_{1j}(\{\mathcal{S}\})^T x + \overline{r}_{1j}(\{\mathcal{S}\}) \geq 0 \ \forall j \in \mathcal{S} \\ & \overline{W}_{1j}(\{\mathcal{S}\})^T x + \overline{r}_{1j}(\{\mathcal{S}\}) \leq 0 \ \forall j \notin \mathcal{S} \\ & \overline{W}(\{\mathcal{S}\})^T x + \overline{r}(\{\mathcal{S}\}) \geq 0. \end{array} \tag{3.30}$$

If the optimal value is negative, we return the verification result $False$ and we collect the solution x^* as a counterexample, denoted as $\hat{x}_{ce}$. 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 3.5. 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 $\lambda^{\mathcal{S}}(x_t)$ and $\xi^{\mathcal{S}}(x_t)$ as follows, when $\mathcal{S}(t) = \mathcal{S}$.

$$\begin{aligned}\lambda^{\mathcal{S}}(x_t) :=& \frac{1}{2}\sum_{j=1}^{M}\left(W_{2j}\tilde{z}(z_{1js})W_{1j}^T - \frac{|W_{2j}|}{R_j}\zeta(x_t)\right)g(x_t)\\ \xi^{\mathcal{S}}(x_t) :=& \frac{1}{2}\sum_{j=1}^{M}\left(W_{2j}\tilde{z}(z_{1js})W_{1j}^T - \frac{|W_{2j}|}{R_j}\zeta(x_t)\right)f(x_t)\\ &- \frac{1}{2}\frac{|W_{2j}|}{R_j}\mathbf{tr}(V_s^T W_{1j} W_{1j}^T V_s).\end{aligned}$$

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 3.4. *Let $\Lambda^{\mathcal{S}}(x_t)$ and $\Xi^{\mathcal{S}}(x_t)$ be as follows.*

$$\Lambda^{\mathcal{S}}(x_t) := \begin{bmatrix}-\lambda^{\mathcal{S}}(x_t)\\ A\end{bmatrix} \qquad \Xi^{\mathcal{S}}(x_t) := \begin{bmatrix}\xi^{\mathcal{S}}(x_t)\\ \mathbf{b}\end{bmatrix}.$$

For all $x \in \tilde{\mathcal{D}}$, there always exists a u satisfying (3.21) if and only if, for all $\mathcal{S} \in \mathbf{Q}$, there is no $x \in \tilde{\mathcal{D}} \cap \mathcal{X}(\{\mathcal{S}\})$ and $\mathbf{y} \in \mathbb{R}^{n_u+1}$ such that

$$[\mathbf{y}]_1 \geq 0, \ \forall i \in \{1, \ldots, n_u + 1\} \tag{3.31a}$$

$$-\mathbf{y}^T \Xi^{\mathcal{S}}(x) < 0 \tag{3.31b}$$

$$\left[\mathbf{y}^T \Lambda^{\mathcal{S}}(x)\right]_1 = 0, \ \forall i \in \{1, \ldots, n_u + 1\}. \tag{3.31c}$$

The proof is similar to Proposition 3.2.

In a similar fashion to the correctness conditions, the feasibility condition (3.31) can be verified by solving the nonlinear program

$$\begin{array}{ll} \min_{x,\mathbf{y}} & \mathbf{y}^T \begin{pmatrix} \xi^{\mathcal{S}}(x) \\ \mathbf{b} \end{pmatrix} \\ \text{s.t.} & \mathbf{y}^T \begin{pmatrix} -\lambda^{\mathcal{S}}(x) \\ A \end{pmatrix} = \mathbf{0} \\ & \mathbf{y} \geq \mathbf{0} \\ & \tilde{B}(x) \geq 0 \\ & \overline{W}_{1j}(\{\mathcal{S}\})^T x + \overline{r}_{1j}(\{\mathcal{S}\}) \geq 0 \ \forall j \in \mathcal{S} \\ & \overline{W}_{1j}(\{\mathcal{S}\})^T x + \overline{r}_{1j}(\{\mathcal{S}\}) < 0 \ \forall j \notin \mathcal{S} \end{array} \tag{3.32}$$

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 x^* as a counterexample, denoted as $\hat{x}_{ce}$.

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 3.2.2. Under the condition of Corollary 3.1, the nonlinear program (3.32) reduces to

$$\begin{array}{ll} \min_{x} & \xi^{\mathcal{S}}(x) \\ \text{s.t.} & \tilde{B}(x) \geq 0 \\ & x \in \mathcal{X}(\{\mathcal{S}\}) \\ & \left[\lambda^{\mathcal{S}}(x)\right]_i = 0, \ \forall i \in \{1, \ldots, n_u\}. \end{array} \tag{3.33}$$

The system with $u = \mathbf{0}$ is feasible in hyperplane $\mathcal{X}(\{\mathcal{S}\})$ if $xi^{\mathcal{S}}(x) \geq 0$. It suffices to verify feasibility by solving the following nonlinear program.

$$\begin{array}{ll} \min_{x} & \xi^{\mathcal{S}}(x) \\ \text{s.t.} & \tilde{B}(x) \geq 0 \\ & x \in \mathcal{X}(\{\mathcal{S}\}). \end{array} \tag{3.34}$$

The ReLU SNCBF verification is summarized in Algorithm. 7. Given an input $x_0 \in \mathcal{D}$, the verification determines if the given ReLU SNCBF B is valid by checking if the conditions in Definition 3.5 are satisfied. If so, the verification returns result $True$, otherwise, it outputs

False and the counterexamples $\hat{x}_{ce}$ where the verification fails. The verification checks if B suffices to satisfy (3.32) by sequentially examining the sufficient conditions (3.34) and (3.33) for better efficiency.

Algorithm 7 ReLU SNCBF Verification

Input: $x_0 \in \mathcal{D}$
Output: Boolean result res, Counterexample $\hat{x}_{ce}$
procedure VERIFICATION($\mathcal{S}(x_0)$)
    $\mathcal{S} \leftarrow$ Enumeration($\mathbf{S}_0$) ▷ Algorithm. 6
    for $\mathcal{S} \in \mathbf{Q}$ **do**
        $res, \hat{x}_{ce} \leftarrow Solve$ (3.30) ▷ Correctness Verification
        $res, \hat{x}_{ce} \leftarrow Solve$ (3.34) ▷ Feasibility Verification
        if $\neg res$ **then**
            $res, \hat{x}_{ce} \leftarrow Solve$ (3.33)
            if $\neg res$ **then**
                $res, \hat{x}_{ce} \leftarrow Solve$ (3.32) ▷ Exact Condition
            end if
        end if
    end for
    Return res, $\hat{x}_{ce}$
end procedure

Synthesis with Verification

The synthesis with verification of ReLU SNCBF shares the same loss function structure (**??**) as the smooth case. For each sample $\hat{x} \in \mathcal{T}$ the safe control signal u is calculated using (3.29). The correctness loss $\mathcal{L}_c(\mathcal{T})$ penalizes the counterexample with negative minima of (3.30) (correctness condition of Definition 3.3) defined as

$$\mathcal{L}_c = \frac{1}{N} \sum_{\hat{x}_i \in \mathcal{X}_s} \max\left(0, \left(\epsilon - \tilde{B}(\hat{x}_i)\right) \mathbb{1}_{\mathcal{X}_s}\right) + \frac{1}{N} \sum_{\hat{x}_j \in \mathcal{X}_U} \max\left(0, \left(\tilde{B}(\hat{x}_j) + \epsilon\right) \mathbb{1}_{\mathcal{X}_U}\right). \quad (3.35)$$

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 (3.31a)-(3.31c) (feasibility condition of Definition 3.5), defined as $\mathcal{L}_f = ||u(\hat{x}) - u_{ref}(\hat{x})||^2 + r$.

3.4 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.

3.4.1 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 present the dynamic models and detailed settings of three cases, namely, the inverted pendulum, the Darboux system, and the unicycle model.

Inverted Pendulum

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

$$d\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \left(\begin{bmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u \right) dt + V\ dv_t, \tag{3.36}$$

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 disturbance $\sigma = \mathsf{diag}(0.1, 0.1)$. 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.

$$\begin{aligned}
\mathcal{X} &= \left\{ x \in \mathbb{R}^2 : x \in \left[-\frac{\pi}{4}, \frac{\pi}{4} \right]^2 \right\} \\
\mathcal{X}_I &= \left\{ x \in \mathbb{R}^2 : x \in \left[-\frac{\pi}{15}, \frac{\pi}{15} \right]^2 \right\} \\
\mathcal{X}_S &= \left\{ x \in \mathbb{R}^2 : x \in \left[-\frac{\pi}{6}, \frac{\pi}{6} \right]^2 \right\} \\
\mathcal{X}_U &= \left\{ x \in \mathbb{R}^2 : x \in \mathcal{X} \backslash \mathcal{X}_S \right\}.
\end{aligned}$$

Darboux

We consider the Darboux system [96], a nonlinear open-loop polynomial system

The dynamic model of Darboux is given as follows.

$$d \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 + 2x_1 x_2 \\ -x_1 + 2x_1^2 - x_2^2 \end{bmatrix} + V \ dv_t. \tag{3.37}$$

We define state space, initial region, and safe region as follows.

$$\begin{aligned}
\mathcal{X} &= \left\{ x \in \mathbb{R}^2 : x \in [-2, 2]^2 \right\} \\
\mathcal{X}_I &= \left\{ x \in \mathbb{R}^2 : x \in [0, 1] \times [1, 2] \right\} \\
\mathcal{X}_S &= \left\{ x \in \mathbb{R}^2 : x_1 + x_2^2 \geq 0 \right\} \\
\mathcal{X}_U &= \left\{ x \in \mathbb{R}^2 : x \in \mathcal{X} \backslash \mathcal{X}_S \right\}
\end{aligned}$$

Unicycle Model

We evaluate our proposed method on a controlled system [97]. We consider vehicles (UAVs) to avoid a pedestrian on the road. The system state consists of a 2-D position and aircraft yaw rate $x := [x_1, x_2, \psi]^T$. The system is manipulated by the yaw rate input u.

$$d \begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix} = \left(\begin{bmatrix} v \cos \psi \\ v \sin \psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \right) dt + V \ dv_t, \tag{3.38}$$

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.

$$\begin{aligned}
\mathcal{X} &= \left\{x \in \mathbb{R}^3 : x \in [-2, 2]^3\right\} \\
\mathcal{X}_I &= \left\{x \in \mathbb{R}^3 : x \in [-0.1, 0.1] \times [-2, -1.8] \times \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]\right\} \\
\mathcal{X}_S &= \left\{x \in \mathbb{R}^3 : \mathcal{X} \setminus \mathcal{X}_U\right\} \\
\mathcal{X}_U &= \left\{x \in \mathbb{R}^2 : x \in [-0.2, 0.2]^2 \times [-2, 2]\right\}.
\end{aligned}$$

3.4.2 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.

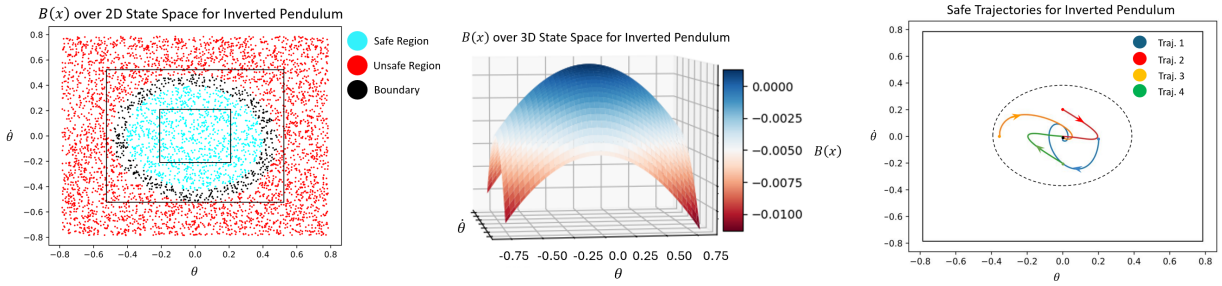
Smooth SNCBF Evaluation

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 $(log(1 + exp(x))$.

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 $B(x)$ with $\psi^* = -0.00042$. Thus, using Theorem 3.3, 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. 3.2a) and in 3D function value heat map (Fig. 3.2b). 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 3.2c shows trajectories initiated inside the safe set (with different reference controllers) never leaving the safe set, thus validating our approach.



(a) (b) (c)

Figure 3.2: This figure presents the experimental results on the inverted pendulum system. Fig. 3.2a visualizes of $B(x)$ 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. 3.2b shows the 3D plot of $B(x)$ over $\mathcal{X}$. Fig. 3.2c presents trajectories initiating inside the safe set following SNCBF-QP, following different reference controllers

Unicycle Model: We then validate our smooth SNCBF-based safe control in the CARLA simulation environment. The vehicle is modeled as the unicycle model 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 $B(x)$ with $\psi^* = -0.04002$.

The trajectory of the vehicle is shown in Fig. 3.3. 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.

ReLU SNCBF Evaluation

We next evaluate the verification-in-the-loop synthesis of ReLU SNCBF.

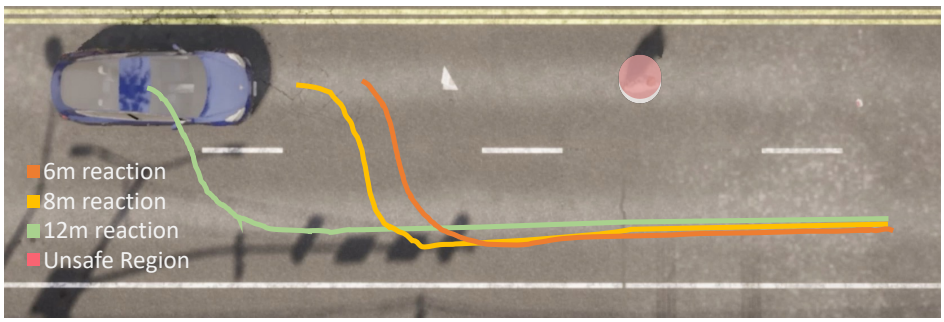


Figure 3.3: Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed SNCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

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-σ-1	3-16-σ-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 3.1: ReLU SNCBF Synthesis Comparison

Table 3.1 summarizes the performance of the proposed ReLU SNCBF approach in verification and synthesis tasks across two distinct scenarios: the Darboux and the unicycle model.

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 3.2: Comparison of Baseline, Smooth and ReLU SNCBF Synthesis. The smooth SNCBF is the proposed verifiable synthesis in Algorithm 5. The ReLU SNCBF is synthesized by VITL with the efficient verifier proposed in Algorithm 7.

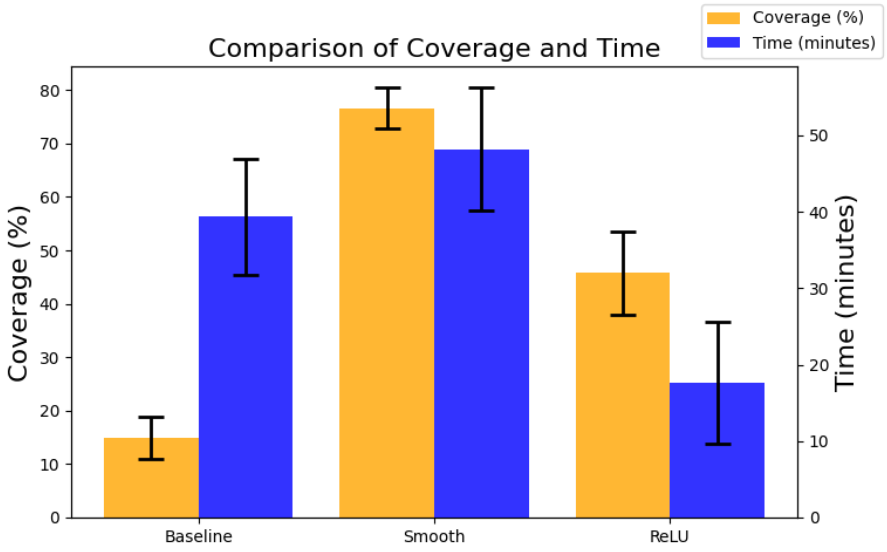


Figure 3.4: Comparison of coverage and training time across Baseline, Smooth and ReLU SNCBF Synthesis. Error bars indicate standard deviations across 5 seeds.

We compare our proposed method with an SNCBF trained with the baseline proposed in [111]. 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 3.2 and Figure 3.4.

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.

3.5 Conclusion

In this paper, we considered the problem of synthesizing a verifiably safe NCBF for stochastic control systems. To address the challenges of synthesizing a verifiably safe SNCBF, we proposed a verification-free synthesis for SNCBF with smooth activation functions. To further address the problem, we extended our framework to a ReLU SNCBF by deriving safety conditions with Tanaka's formula for a single hidden layer ReLU SNCBF and proposing efficient verification algorithms. We proposed verifiers for VITL synthesis for SNCBFs with smooth activation functions and ReLU activation functions. We validated our synthesis and verification frameworks in three cases, namely, the inverted pendulum, Darboux, and the unicycle model. The results of the experiment illustrated the improvement in efficiency and coverage compared to the baseline method and demonstrated the advantages and disadvantages of using a smooth SNCBF or a ReLU SNCBF.

Limitations: The paper presented a comprehensive study of NCBFs for stochastic systems with smooth and ReLU activation functions for a single hidden layer. However, the synthesis and verification of multi-hidden-layer SNCBFs with ReLU activation functions remains an open problem, and we will study this in future work.

Chapter 4

Resilient Safe Control under Low-Dimensional Sensor Faults

Sensor faults and attacks may cause errors in the sensor measurements and system dynamics, which leads to erroneous control inputs and hence safety violations. In this chapter, we improve the robustness against sensor faults and attacks by proposing a class of Fault-Tolerant Control Barrier Functions (FT-CBFs) for nonlinear systems. Our approach maintains a set of state estimators according to fault patterns and incorporates CBF-based constraints to ensure safety under sensor faults. We then propose a framework for joint safety and stability by integrating FT-CBFs with Control Lyapunov Functions. We propose a sum-of-squares (SOS) based approach to verify the feasibility of FT-CBFs for both sensor faults. We evaluate our approach via case study on a wheeled mobile robot (WMR) system in the presence of a sensor attack.

Inspired by the universal approximation power of neural networks, there is a growing trend toward representing CBFs using neural networks, leading to the notion of neural CBFs (NCBFs). Current NCBFs, however, are trained and deployed in benign environments, making them ineffective for scenarios where robotic systems experience sensor faults and attacks. We derive the necessary and sufficient conditions for FT-NCBFs to guarantee safety, and develop a data-driven method to learn FT-NCBFs by minimizing a loss function constructed using the derived conditions. Using the learned FT-NCBF, we synthesize a control input and formally prove the safety guarantee provided by our approach. We demonstrate our proposed approach using two case studies: obstacle avoidance problem for an autonomous mobile robot and spacecraft rendezvous problem

This chapter made the following contributions:

- We propose High-order Stochastic CBFs (HOSCBF) for the system with high relative degree and propose FT-SCBFs with high order degree to ensure finite time safety when sensor faults occur. We propose an SOS-based scheme to verify the feasibility of constraints of FT-SCBFs with high relative degree.
- We compose HOSCBFs with Control Lyapunov Functions (CLFs) to provide joint guarantees on safety and stability under sensor faults.
- We evaluate our approach on a wheeled mobile robot (WMR) system in the presence of a sensor attack. The proposed HOSCBF-CLF ensures safety and convergence of a wheeled mobile robot (WMR) system in the presence of a sensor attack.
- We propose FT-NCBFs for robotic systems under sensor faults and attacks. We derive the necessary and sufficient conditions for FT-NCBFs to guarantee safety. Based on the derived conditions, we develop a data-driven method to learn FT-NCBFs.
- We develop a fault-tolerant framework which utilizes our proposed FT-NCBFs for safety-critical control synthesis. We prove that the synthesized control inputs guarantee safety under all fault and attack patterns.
- We evaluate our approach using two case studies on the obstacle avoidance problem of a mobile robot and the spacecraft rendezvous problem. We show that our approach guarantees the robot to satisfy the safety constraint regardless of the faults and attacks, whereas the baseline employing the existing NCBFs fails.

The remainder of this chapter is organized as follows. Section 4.1 presents the related work. Section 4.2 presents background and preliminaries. Section 4.3 proposes a HOSCBF-based control policy for systems under sensor faults and attacks as well as a scheme to verify the feasibility of HOSCBFs. Section 4.4 proposes a framework for joint safety and stability via HOSCBF-CLFs. Section 4.5 presents Fault Tolerant NCBF. Section 4.6 concludes the chapter.

4.1 Related Work

Fault-tolerant control systems (FTCS) aim to accommodate faults and maintain stability of the system with little or acceptable degradation in performance. See [45] for an in-depth

treatment. FTCS are classified into two main types, namely, active FTCS and passive FTCS [46]. In active FTCS, Fault Detection and Isolation (FDI) plays a significant role and has been studied for decades. See [112] for more details.

Active FTCS against sensor faults and attacks include statistical hypothesis testing for stochastic systems [113], and unknown input observers for deterministic systems [114]. Kalman Filter (KF) and Extended Kalman Filter (EKF) are extensively used in FDI applications such as [115] for MIMO system. More recently, data-driven approaches to fault tolerance have shown promise [116, 117]. While the approach of using Kalman filter residues to identify potential faults is related to our conflict resolution approach, safety of the system under faults and attacks is not addressed. Sliding-mode control, as one of popular PFTCS, is proposed for singularly perturbed systems [118], switched systems [119], fuzzy systems, [120], and Markov Jump Systems [121, 122]. Several of these works aim to guarantee stability in the presence of faults [123], which is related to but distinct from the safety criteria we consider.

Passive FTCS against actuator failures is proposed due to its advantage of fast response. Reliable control for Linear time-invariant (LTI) system under actuator failure is proposed in [124] and implemented for LTI aircraft model in [125].

Countermeasures incorporating disturbance observer-based CBF are proposed to ensure robust safety of systems with model uncertainties [47, 48] and model-free safe reinforcement learning [49]. With the growing attention on faults and attacks, safety guarantees on systems under faulty components or adversarial environments have become an active research area. Existing FTC approaches focus on maintaining performance and do not provide provable safety guarantees. Barrier certificate based fault-tolerant Linear quadratic Gaussian (LQG) tracking is investigated in [55] for LTI system under sensor fault and false data injection attack and generalized to multiple possible compromised sensor sets in [126]. Compared with barrier certificate method, CBF-based approaches have more advantages on flexibility.

4.2 Preliminaries

In this section, we present the system model and provide background on the EKF and CBFs.

4.2.1 System Model

Notations. For a set S, we denote $\text{int}(S)$ and ∂S as the interior and boundary of S, respectively. For any vector v, we let $[v]_i$ denote the i-th element of v. We let $\overline{\lambda}(A)$ denote the magnitude of the largest eigenvalue of matrix A, noting that this is equal to the largest eigenvalue when A is symmetric and positive definite. When the value of A is clear, we write $\overline{\lambda}$.

We consider a nonlinear control system with state $x_t \in \mathbb{R}^n$ and input $u_t \in \mathbb{R}^p$ at time t. The state dynamics and the system output $y_t \in \mathbb{R}^q$ are described by the stochastic differential equations

$$dx_t = (f(x_t) + g(x_t)u_t)\ dt + \sigma_t\ dW_t \tag{4.1}$$

$$dy_t = cx_t\ dt + \nu_t\ dV_t \tag{4.2}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times p}$ are locally Lipschitz, $\sigma_t \in \mathbb{R}^{n\times n}$, W_t is an n-dimensional Brownian motion, $c \in \mathbb{R}^{q\times n}$, $\nu_t \in \mathbb{R}^{q\times q}$, and V_t is a q-dimensional Brownian motion.

The safety conditions of a system are specified in terms of forward invariance of a pre-defined safe region. We define the safe region as follows.

Definition 4.1 (Safe Region)**.** *The safe region of the system is a set $\mathcal{C} \subseteq \mathbb{R}^n$ defined by*

$$\mathcal{C} = \{x : h(x) \geq 0\}, \quad \partial\mathcal{C} = \{x : h(x) = 0\} \tag{4.3}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is twice-differentiable on $\mathcal{C}$.

We assume throughout the paper that $x_0 \in \text{int}(\mathcal{C})$, i.e., the system is initially safe. Let $\overline{f}(x,u) = f(x) + g(x)u$. The uniform detectability property is defined as follows.

Definition 4.2 (Uniform Detectability)**.** *The pair $[\frac{\partial \overline{f}}{\partial x}(x,u), c]$ is uniformly detectable if there exists a bounded, matrix-valued function $\Theta(x)$ and a real number $\eta > 0$ such that*

$$w^T \left(\frac{\partial \overline{f}}{\partial x}(x,u) + \Theta(x)c \right) w \leq -\eta ||w||^2$$

for all w, u, and x.

4.2.2 Background and Preliminary Results

Let $\hat{x}_t$ denote the EKF estimate of x_t. The EKF for the system described by (4.1) and (4.2) is defined by

$$d\hat{x}_t = (f(\hat{x}_t) + g(\hat{x}_t)u_t)dt + K_t(dy_t - c\hat{x}_t),$$

where $K_t = P_t c^T R_t^{-1}$ and $R_t = \nu_t \nu_t^T$. The matrix P_t is the positive-definite solution to

$$\frac{dP}{dt} = F_t P_t + P_t F_t^T + Q_t - P_t c^T R_t^{-1} c P_t$$

where $Q_t = \sigma_t \sigma_t^T$ and $F_t = \frac{\partial \overline{f}}{\partial x}(\hat{x}_t, u_t)$. To utilize EKF, we make the following assumptions.

Assumption 4.1. *The SDEs (4.1) and (4.2) satisfy the conditions:*

1. *There exist constants β_1 and β_2 such that $\mathbf{E}(\sigma_t \sigma_t^T) \geq \beta_1 I$ and $\mathbf{E}(\nu_t \nu_t^T) \geq \beta_2 I$ for all t.*
2. *The pair $[\frac{\partial \overline{f}}{\partial x}(x, u), c]$ is uniformly detectable.*
3. *Let ϕ be defined by*

$$\overline{f}(x, u) - \overline{f}(\hat{x}, u) = \frac{\partial \overline{f}}{\partial x}(x - \hat{x}) + \phi(x, \hat{x}, u).$$

Then there exist real numbers k_ϕ and ϵ_ϕ such that

$$||\phi(x, \hat{x}, u)|| \leq k_\phi ||x - \hat{x}||_2^2$$

for all x and $\hat{x}$ satisfying $||x - \hat{x}||_2 \leq \epsilon_\phi$.

One can obtain k_ϕ by considering a compact subset $\kappa \subseteq \mathbb{R}$. For instance, given a function f that is twice differentiable with respect to x in subset κ, we have

$$k_\phi = \max_{1 \leq i \leq n} \sup_{x \in \kappa} \|\nabla^2 f_i(x, u)\|,$$

where $\nabla^2 f$ is the Hessian matrix. The bounds on estimation error ϵ_ϕ can be calculated via an integral formula [127, Chapter 20]. More details can be found in [128]. The following result describes the accuracy of the EKF.

Theorem 4.1 ([128]). *Suppose that the conditions of Assumption 4.1 hold. Then there exists* $\delta > 0$ *such that* $\sigma_t \sigma_t^T \leq \delta I$ *and* $\nu_t \nu_t^T \leq \delta I$. *For any* $0 < \epsilon < 1$, *there exists* $\gamma > 0$ *such that*

$$Pr\left(\sup_{t \geq 0} ||x_t - \hat{x}_t||_2 \leq \gamma\right) \geq 1 - \epsilon.$$

We next provide background and preliminary results on stochastic CBFs. The following theorem provides sufficient conditions for safety of a stochastic system. Given a finite time T, suppose the mapping h is a continuous function with linear function kx as the class-κ function, where $k \geq 0$. Let the control input u_t be chosen to satisfy

$$\frac{\partial h}{\partial x}(f(x_t) + g(x_t)u_t) + \frac{1}{2} tr\left(\sigma_t^T \frac{\partial^2 h}{\partial x^2}(x_t)\sigma_t\right) \geq -kh(x_t). \tag{4.4}$$

Let $\zeta = \sup_{x \in \mathcal{C}} h(x)$ and $x_0 \in int(C)$. By Proposition 3.1 we have

$$Pr(x_t \in int(\mathcal{C}), 0 \leq t \leq T) \geq (\frac{h(x_0)}{\zeta})e^{-\zeta T}.$$

Theorem 4.2. *For a system (1)–(2) with safety region defined by (3), define*

$$\overline{h}_\gamma = \sup_{x, x^0} \{h(x) : ||x - x^0||_2 \leq \gamma \text{ and } h(x^0) = 0\},$$

where $\hat{h}(x) := h(x) - \overline{h}_\gamma$. *Let* $\frac{\partial h}{\partial x}$ *denote the derivative* $\frac{\partial h}{\partial x}|_{x=\hat{x}}$ *for simplicity. Let* $z_t \in \mathbb{R}^n$ *represent the estimation error, where* $z_t = (x_t - \hat{x}_t)$. *Suppose that* u_t *is chosen to satisfy*

$$\frac{\partial h}{\partial x}(f(\hat{x}_t) + g(\hat{x}_t)u_t) - \left\|\frac{\partial h}{\partial x} K_t c\right\|_2 \gamma + \frac{1}{2} tr\left(\nu_t^T K_t^T \frac{\partial^2 h}{\partial x^2}(\hat{x}_t) K_t \nu_t\right) \geq -\hat{h}(\hat{x}_t). \tag{4.5}$$

Then $Pr(x_t \in \mathcal{C}, 0 \leq t \leq T| \ ||x_t - \hat{x}_t||_2 \leq \gamma) \geq (\frac{\hat{h}(x_0)}{\zeta})e^{-\zeta T}$.

Proof. We have the estimate $\hat{x}$ yields

$$\begin{aligned} d\hat{x}_t &= \bar{f}\left(\hat{x}_t, u_t\right) dt + K_t \left(c x_t dt + \nu_t dV_t - c\hat{x}_t dt\right) \\ &= \left(\bar{f}\left(\hat{x}_t, u_t\right) + K_t c\left(x_t - \hat{x}_t\right)\right) dt + K_t \nu_t dV_t \end{aligned}$$

Given $||x_t - \hat{x}_t||_2 \leq \gamma$, we have

$$\frac{\partial \hat{h}}{\partial x} K_t c \left(x_t - \hat{x}_t\right) \geq - \left\| \frac{\partial \hat{h}}{\partial x} K_t c \right\|_2 \left\| z_t \right\|_2 \geq - \left\| \frac{\partial \hat{h}}{\partial x} K_t c \right\|_2 \gamma$$

We then choose u_t to satisfy (4.5). Then, we have

$$\begin{aligned} &\frac{\partial \hat{h}}{\partial x} \left(f\left(\hat{x}_t\right) + g\left(\hat{x}_t\right) u_t + K_t c \left(x_t - \hat{x}_t\right)\right) + \\ &\qquad \frac{1}{2} \operatorname{tr}\left(\nu_t^T K_t^T \left(\frac{\partial^2 \hat{h}}{\partial x^2}\right) K_t \nu_t\right) + \hat{h}(\hat{x}_t) \geq \\ &\frac{\partial h}{\partial x}(f(\hat{x}_t) + g(\hat{x}_t) u_t) - \left\| \frac{\partial h}{\partial x} K_t c \right\|_2 \gamma + \\ &\qquad \frac{1}{2} tr\left(\nu_t^T K_t^T \frac{\partial^2 h}{\partial x^2}(\hat{x}_t) K_t \nu_t\right) + \hat{h}(\hat{x}_t) \geq 0 \end{aligned}$$

Hence, by Proposition 1, we have $Pr(x_t \in \mathcal{C}, 0 \leq t \leq T |\ ||x_t - \hat{x}_t||_2 \leq \gamma) \geq (\frac{\hat{h}(x_0)}{\zeta}) e^{-\zeta T}$. $\square$

Intuitively, Eq. (4.5) implies that as the state approaches the boundary, the control input is chosen such that the rate of increase of the barrier function decreases to zero. Hence Theorem 4.2 implies that if there exists an SCBF for a system, then the safety condition is satisfied with probability greater or equal to $(1 - \epsilon)(\frac{\hat{h}(x_0)}{\zeta}) e^{-\zeta T}$ when an EKF is used as an estimator and the control input is chosen at each time t to satisfy (4.5). We next present a probabilistic guarantee for HOCBFs within a finite time horizon.

Definition 4.3 (SCBF)**.** *The function h is a* Stochastic Control Barrier Function (SCBF) *of the system, if for all $\hat{x}_t$ satisfying $h(\hat{x}_t) > 0$ there exists u_t s.t. $\forall z_t$ with $\|z_t\| \leq \gamma$ (4.5) is satisfied.*

There may not exist a value of u_t to satisfy (4.5) when $\frac{\partial h}{\partial x} g(x) = 0$. To address this problem, CBF with high relative degree has been proposed in deterministic [20] and stochastic [44] settings.

Let $d = 0, 1, \ldots$ and define d^{th} order differentiable function $h^d(x)$ as

$$\begin{aligned} h^0(x) &= h(x), \\ h^1(x) &= \frac{\partial h^0}{\partial x}\overline{f}(x,u) + \frac{1}{2}\operatorname{tr}\left(\sigma^T\left(\frac{\partial^2 h^0}{\partial x^2}\right)\sigma\right) + h^0(x), \\ &\vdots \\ h^{d+1}(x) &= \frac{\partial h^d}{\partial x}\overline{f}(x,u) + \frac{1}{2}\operatorname{tr}\left(\sigma^T\left(\frac{\partial^2 h^d}{\partial x^2}\right)\sigma\right) + h^d(x). \end{aligned}$$

Define $\mathcal{C}^d = \left\{x : h^d(x) \geq 0\right\}$. The following theorem provides sufficient conditions for safety of a high-degree system.

Theorem 4.3. *Let $\overline{\mathcal{C}} = \bigcap_{d=0}^{d'} \mathcal{C}^d$. Suppose that there exists d' such that $\frac{\partial h^{d'}}{\partial x} g(x)u \neq 0$. If u_t is chosen to satisfy*

$$\frac{\partial h^{d'}}{\partial x}(\overline{f}(x,u)) + \frac{1}{2}\operatorname{tr}\left(\sigma^T \frac{\partial^2 h^{d'}}{\partial x^2}\sigma\right) \geq -h^{d'}(x). \tag{4.6}$$

Let $\zeta^d = \sup_{x\in\mathcal{C}^d} h^d(x)$. Then $Pr(x_t \in \mathcal{C}, 0 \leq t \leq T) \geq \prod_{d=0}^{d'}(\frac{h^d(x_0)}{\zeta^d})e^{-\zeta^d T}$ if $x_0 \in \overline{\mathcal{C}}$.

Proof. Suppose that u_t satisfying the conditions of the theorem is chosen at each time t. Theorem 2 implies that $h^{d'}(x_t) \geq 0$ when $0 \leq t \leq T$ with probability greater or equal to $(\frac{h^{d'}(x_0)}{\zeta^{d'}})e^{-\zeta^{d'}T}$. By definition of $h^d(x)$ we have that $\frac{\partial h^{d'-1}}{\partial x} g(x)u = 0$. We also have $Pr(x_t \in \mathcal{C}^{d'-1}, 0 \leq t \leq T | x_t \in \mathcal{C}^{d'}) \geq (\frac{h^{d'-1}(x_0)}{\zeta^{d'-1}})e^{-\zeta^{d'-1}T}$. Proceeding inductively, we have $Pr(x_t \in \mathcal{C}, 0 \leq t \leq T) \geq P$ where

$$P = \prod_{i=1}^{d'} Pr(x_t \in \mathcal{C}^{i-1}, 0 \leq t \leq T \mid x_t \in \mathcal{C}^i) \cdot Pr(x_t \in \mathcal{C}^{d'}, 0 \leq t \leq T).$$

Finally, we have $Pr(x_t \in \mathcal{C}, 0 \leq t \leq T) \geq \prod_{d=0}^{d'}(\frac{h^d(x_0)}{\zeta^d})e^{-\zeta^d T}$. □

Definition 4.4. *The function $h^{d'}$ is a high-order CBF (HOCBF) of relative degree d' for system* (4.1)-(4.2) *if for all $x \in \overline{\mathcal{C}}$ there exists u satisfying* (4.6).

4.3 Safe Control Under Sensor Faults and Attacks

In this section, we consider the system under sensor faults and derive safety guarantees. We first present the overview of our approach, followed by detailed formulation of faults and fault-tolerant control framework.

4.3.1 Overview of the Approach

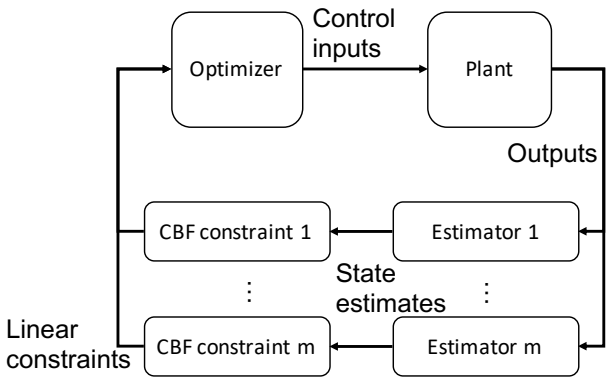


Figure 4.1: Schematic illustration of our proposed approach for system under sensor faults and attacks.

To ensure safety of nonlinear systems under sensor faults and attacks, we propose a class of CBFs, which is shown in Figure 4.1 and constructed as follows. We maintain a set of state estimators, each omitting a set of sensors associated with one fault pattern and then use CBF constraints to ensure that each of the estimated states remains within the safe region. The intuition of our approach is that one can simply ignore the faulty measurements by omitting the sensors of each fault pattern and construct CBFs for each estimate. However, it may be infeasible to satisfy all CBF constraints using a single control input when faults occur and the state estimates deviate due to the fault. To resolve conflicts, we propose a threshold-based method to exclude outlier estimates and relax the corresponding CBF constraints. Since the index of the faulty sensor is unknown, the state estimates may not be consistent. We call the inconsistency 'conflicts'. To resolve conflicts, we maintain a second set of estimators omitting sensors affected by conflicting fault patterns and propose a threshold-based method to relax Given two constraints that conflict with each other, we compare each state estimate to the corresponding estimator that excludes all sensors affected by both fault patterns. If

the difference between the conflicting estimate and the estimate excluding all sensors affected by both fault patterns exceeds a given threshold, we relax the corresponding constraint.

4.3.2 Sensor Fault Pattern Formulation

We consider a nonlinear control system whose output may be affected by one of m sensor faults. The set of possible faults is indexed as $\{r_1, \ldots, r_m\}$. Each fault r_i maps to a set of affected observations $\mathcal{F}(r_i) \subseteq \{1, \ldots, q\}$. We assume that $\mathcal{F}(r_i) \cap \mathcal{F}(r_j) = \emptyset$ for $i \neq j$. Let $r \in \{r_1, \ldots, r_m\}$ denote the index of the fault experienced by the system. The system dynamics are described as

$$dx_t = (f(x_t) + g(x_t)u_t)\ dt + \sigma_t\ dW_t \tag{4.7}$$

$$dy_t = (cx_t + a_t)\ dt + \nu_t\ dV_t, \tag{4.8}$$

where the vector $a_t \in \mathbb{R}^q$ represents the impact of the fault and is constrained by $\text{supp}(a_t) \subseteq \mathcal{F}(r)$. Hence, if fault r_i occurs, then the outputs of any of the sensors indexed in $\mathcal{F}(r_i)$ can be arbitrarily modified by the fault. The sets $\mathcal{F}(r_1), \ldots, \mathcal{F}(r_m)$ are known, but the value of r_i is unknown. In other words, the set of possible faults is known, but the exact fault that has occurred is unknown to the controller.

To illustrate, we take an autonomous vehicle system as an example. Consider an autonomous system equipped with two INS sensors, a GNSS and one LiDAR system indexed as $\{1, 2, 3, 4\}$ for localization measurements denoted as $y = \{y_1, y_2, y_3, y_4\}$. We consider three possible attacks: an attack on one of the INS sensors, an attack on another INS sensor, or a simultaneous GPS/LiDAR spoofing attack. The corresponding fault patterns are given as $\mathcal{F}(r_1) = \{1\}$, $\mathcal{F}(r_2) = \{2\}$, $\mathcal{F}(r_3) = \{3, 4\}$.

Define $\overline{c}_i$ to be the c matrix with the corresponding rows indexed in $\mathcal{F}(r_i)$ removed, $\overline{y}_{t,i}$ to be equal to the vector y_t with the entries indexed in $\mathcal{F}(r_i)$ removed, and $\overline{\nu}_{t,i}$ to be the matrix ν_t with rows and columns indexed in $\mathcal{F}(r_i)$ removed. Define $\overline{c}_{i,j}$ to be the c matrix with the corresponding rows indexed in $\mathcal{F}(r_i)$ and $\mathcal{F}(r_j)$ removed, where $i \neq j$.

We make the following assumption for the sensor fault scenario.

Assumption 4.2. *The system* (4.7)-(4.8) *and the sensor fault patterns* $\mathcal{F}(r_1), \ldots, \mathcal{F}(r_m)$ *satisfy the conditions:*

1. *The system is controllable.*
2. *For each* $i, j \in \{1, \ldots, m\}$*, the pair* $[\frac{\partial \overline{f}}{\partial x}(x, u), \overline{c}_{i,j}]$ *is uniformly detectable.*

Problem Statement: Given a finite time T, a safe set $\mathcal{C}$ defined in (4.3) and a parameter $\epsilon \in (0, 1)$, compute a control policy that, at each time t, maps the sequence $\{y_{t'} : t' \in [0, t)\}$ to an input u_t such that, for any fault $r \in \{r_1, \ldots, r_m\}$, $Pr(x_t \in \mathcal{C},\ 0 < t < T) \geq (1 - \epsilon)\mathcal{T}(T)$, for some function $\mathcal{T} : \mathbb{R}^n \to (0, 1)$.

4.3.3 Sensor FTC Strategy Definition

We propose a CBF-based strategy with safety guarantees for a system satisfying Assumption 4.2. The strategy that accommodates sensor faults and attacks is an FTC in a passive manner. The goal of FTC is to ensure the robustness of the control system to accommodate multiple component faults without striving for optimal performance for any specific fault condition.

The intuition behind our approach is as follows. Since we do not know the fault pattern r, we construct estimators excluding faulty sensors by maintaining m EKFs. Each EKF corresponds to a different possible fault pattern in $\{r_1, \ldots r_m\}$. We ensure safety with desired probability by defining m corresponding SCBFs, each of which results in a different linear constraint on the control input.

The potential drawback is that the safety guarantees of Theorem 4.2 rely on the existence of a control input satisfying the safety constraint at each time-step. This assumption may not hold for two reasons. Firstly, feasible control input u may not exist when $\frac{\partial h}{\partial x} g(x) = 0$, since u does not affect states x. Secondly, a feasible solution may not exist when faulty sensor measurements cause the state estimates to diverge. To address the first reason, we define higher-order SCBFs such that for d-th degree $\frac{\partial h^d}{\partial x} g(x) \neq 0$. Then, we choose control input u to satisfy constraints constructed by higher-order SCBFs. To address the second reason, we define a set of $\binom{m}{2}$ EKFs to resolve conflicts between the constraints. Each EKF estimator

omits all sensors affected by either fault r_i or fault r_j for some $i, j \in \{1, \ldots, m\}$, $i \neq j$. These estimators will be used to resolve any deviations between the state estimates from sensors $\{1, ..., m\} \backslash \mathcal{F}(r_i)$ and $\{1, ..., m\} \backslash \mathcal{F}(r_j)$.

Let $\mathcal{C}_\gamma := \{x : \hat{h}^d(x) \geq 0\}$ where $\hat{h}^d(x) = h^d(x) - \bar{h}^d_\gamma$ and

$$\bar{h}^d_\gamma = \sup_{x, x^{d,0}} \left\{ h^d(x) : \|x - x^{d,0}\|_2 \leq \gamma \text{ and } h^d(x^{d,0}) = 0 \right\} \tag{4.9}$$

Let $\overline{\mathcal{C}}_\gamma = \bigcap_{d=0}^{d'} \mathcal{C}^d_\gamma$. To ensure safety as defined in (4.3), we need to show that $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T) \geq \prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$ if $x_0 \in \bar{\mathcal{C}}_\gamma$ given $x_0 \in \bar{\mathcal{C}}_\gamma$ and $||x_t - \hat{x}_t||_2 \leq \gamma,\ \forall t$.

Proposition 4.1. *For a system* (4.7)-(4.8) *with safety region defined by* (4.3)*, suppose there exists d', such that $\frac{\partial h^{d'}}{\partial x} g(x) \neq 0$. Suppose that u_t is chosen to satisfy*

$$\frac{\partial h^{d'}}{\partial x} \bar{f}(\hat{x}_t, u_t) - \left\| \frac{\partial h^{d'}}{\partial x}(\hat{x}_t) K_t c \right\|_2 \gamma + \frac{1}{2} tr \left(\nu_t^T K_t^T \frac{\partial^2 h^{d'}}{\partial x}(\hat{x}_t) K_t \nu_t \right) \geq -\hat{h}^{d'}(\hat{x}_t). \tag{4.10}$$

Then $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T |\ ||x_t - \hat{x}_t||_2 \leq \gamma\ \forall t) \geq \prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$ if $x_0 \in \bar{\mathcal{C}}_\gamma$.

Proof. Given $||x_t - \hat{x}_t||_2 \leq \gamma\ \forall t$, we suppose that u_t is chosen to satisfy (4.10). By Theorem 4.2 and (4.10), we have $h^{d'}(x_t) \geq 0$ when $0 \leq t \leq T$ with probability greater or equal to $(\frac{\hat{h}^{d'}(x_0)}{\zeta^{d'}}) e^{-\zeta^{d'} T}$. By definition of relative degree, we have $\frac{\partial h^d}{\partial x} g(x) u = 0$ for $d < d'$. By definition of $h^{d'}(x_t)$, we have $\frac{\partial h^d}{\partial x} \overline{f}(x, 0) + \frac{1}{2} \operatorname{tr} \left(\sigma^T \left(\frac{\partial^2 h^d}{\partial x^2} \right) \sigma \right) + h^d(x) \geq 0$, where $d = d' - 1$. This implies $h^{d'-1}(x_t) \geq 0$. Similar to the proof of Theorem 4.3, by proceeding inductively, we then have $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T |\ ||x_t - \hat{x}_t||_2 \leq \gamma\ \forall t) \geq \prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$ if $x_0 \in \bar{\mathcal{C}}_\gamma$. □

The function $h^{d'}$ ensures safety of the system with relative degree d' by Proposition 4.1. Hence we define the function as follows.

Definition 4.5. *The function $h^{d'}$ is a* higher order SCBF (HOSCBF) *of relative degree d' for system* (4.1)-(4.2) *if for all $\hat{x}_t \in \overline{\mathcal{C}}$ there exists u_t satisfying* (4.10).

We next present a scheme to resolve conflicts between constraints in the case of faults and attacks. Let $\overline{R}_{t,i} = \overline{\nu}_{t,i} \overline{\nu}^T_{t,i}$ and $K_{t,i} = \overline{P}_{t,i} \overline{c}^T_i (\overline{R}_{t,i})^{-1}$. Here $\overline{P}_{t,i}$ is the solution to the Riccati

differential equation

$$\frac{d\overline{P}_{t,i}}{dt} = F_{t,i}\overline{P}_{t,i} + \overline{P}_{t,i}F_{t,i}^T + Q_t - \overline{P}_{t,i}\overline{c}_i^T\overline{R}_{t,i}^{-1}\overline{c}_i\overline{P}_{t,i}$$

with $F_{t,i} = \frac{\partial \overline{f}}{\partial x}(\hat{x}_{t,i}, u_t)$. Define a set of m EKFs with estimates denoted $\hat{x}_{t,i}$ via

$$d\hat{x}_{t,i} = (f(\hat{x}_{t,i}) + g(\hat{x}_{t,i})u_t)\ dt + K_{t,i}(d\overline{y}_{t,i} - \overline{c}_i\hat{x}_{t,i}\ dt). \tag{4.11}$$

Each of these EKFs represents the estimate obtained by removing the sensors affected by fault r_i. Furthermore, define $\overline{y}_{t,i,j}$, $\overline{\nu}_{t,i,j}$, $\overline{c}_{i,j}$, $\overline{R}_{t,i,j}$, and $K_{t,i,j}$ in an analogous fashion with entries indexed in $\mathcal{F}(r_i) \cup \mathcal{F}(r_j)$ removed. We assume throughout that the $\overline{R}$ matrices are invertible. We then define a set of $\binom{m}{2}$ estimators $\hat{x}_{t,i,j}$ as

$$d\hat{x}_{t,i,j} = (f(\hat{x}_{t,i,j}) + g(\hat{x}_{t,i,j})u_t)\ dt + K_{t,i,j}(d\overline{y}_{t,i,j} - \overline{c}_{i,j}\hat{x}_{t,i,j}\ dt). \tag{4.12}$$

When $\mathcal{F}(r_i) \cup \mathcal{F}(r_j) = \{1, \ldots, q\}$, the open-loop estimator is used for $\hat{x}_{t,i,j}$.

We then select parameters $\gamma_1, \ldots, \gamma_m \in \mathbb{R}_+$, and $\{\theta_{ij} : i < j\} \subseteq \mathbb{R}_+$. The set of feasible control inputs is defined at each time t using the following steps:

1. Define $Z_t = \{1, \ldots, m\}$. Define a collection of sets Ω_i, $i \in Z_t$, by

$$\Omega_i \triangleq \left\{ u : \frac{\partial h_i^{d'}}{\partial x}\left(\bar{f}(\hat{x}_{t,i}, u_t)\right) - \left\| \frac{\partial h_i^{d'}}{\partial x}(\hat{x}_t)K_t c \right\|_2 \gamma_i + \frac{1}{2} tr\left(\nu_t^T K_t^T \frac{\partial^2 h_i^{d'}}{\partial x}(\hat{x}_{t,i})K_t\nu_t\right) \geq -\hat{h}_i^{d'}(\hat{x}_{t,i}) \right\} \tag{4.13}$$

   Select u_t satisfying $u_t \in \bigcap_{i \in Z_t} \Omega_i$. If no such u_t exists, there exists conflicts between constraints, i.e., $\exists i, j$, $i \neq j$ s.t. $\Omega_i \cap \Omega_j = \emptyset$. Then go to Step 2.

2. For each i, j with $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 > \theta_{ij}$, set $Z_t = Z_t \setminus \{i\}$ (resp. $Z_t = Z_t \setminus \{j\}$) if $||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 > \theta_{ij}/2$ (resp. $||\hat{x}_{t,j} - \hat{x}_{t,i,j}||_2 > \theta_{ij}/2$). If $\bigcap_{i \in Z_t} \Omega_i \neq \emptyset$, then select $u_t \in \bigcap_{i \in Z_t} \Omega_i$. Else, go to Step 3. This step resolves conflicts between estimations by comparing the difference between estimations against thresholds θ_{ij}.

3. Remove the indices i from Z_t corresponding to the estimators with the largest residue values $\overline{y}_{t,i} - \overline{c}_i \hat{x}_{t,i}$ until there exists $u_t \in \bigcap_{i \in Z_t} \Omega_i$.

We next provide sufficient conditions for this control policy to guarantee safety.

Theorem 4.4. *Given $x_0 \in \bar{\mathcal{C}}_\gamma$, define*

$$\overline{h}^d_{\gamma_i} = \sup_{x, x^0} \left\{ h^d(x) : ||x - x^{d,0}||_2 \leq \gamma_i \quad \textit{and } h^d(x^{d,0}) = 0 \right\}$$

and $\hat{h}^d_i(x) = h^d(x) - \overline{h}^d_{\gamma_i}$. Suppose $\gamma_1, \ldots, \gamma_m$, and θ_{ij} for $i < j$ are chosen such that the following conditions are satisfied:

1. *Define $\Lambda^d_i(\hat{x}_{t,i}) = \frac{\partial h^d_i}{\partial x}(\hat{x}_{t,i}) g(\hat{x}_{t,i})$. For all $i, j \in Z_t$ with $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 \leq \theta_{ij}$, there exists u such that*

$$\Lambda^{d'}_i(\hat{x}_{t,i}) u > 0 \tag{4.14}$$

   for all $i \in Z'_t$.

2. *For each i, when $r = r_i$,*

$$Pr(||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 \leq \frac{\theta_{ij}}{2} \ \forall j, \\ ||\hat{x}_{t,i} - x_t||_2 \leq \gamma_i \ \forall t) \geq 1 - \epsilon. \tag{4.15}$$

Then $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T) \geq (1-\epsilon) \prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$ for any fault pattern $r \in \{r_1, \ldots, r_m\}$.

Proof. Suppose that the fault $f = f_i$. We will show that, if $||\hat{x}_{t,i} - x_t||_2 \leq \gamma_i$ and $||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 \leq \theta_{ij}/2$ for all t, then $u_t \in \Omega_i$ holds. Hence $x_t \in \mathcal{C}$ for $0 \leq t \leq T$ with probability greater or equal to $\prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$ by Proposition 4.1.

At time t, suppose that $\hat{h}^{d'}_i(\hat{x}_{t,i}) \geq 0$, and that $||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 \leq \theta_{ij}/2$. We consider three cases, namely (i) $||\hat{x}_{t,j} - \hat{x}_{t,k}||_2 \leq \theta_{jk}$ for all $j, k \in Z_t$, (ii) $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 \leq \theta_{ij}$ for all $j \in Z_t$, but there exist $j, k \in Z_t \setminus \{i\}$ such that $||\hat{x}_{t,j} - \hat{x}_{t,k}||_2 > \theta_{jk}$, and (iii) $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 > \theta_{ij}$ for some $j \in Z_t$.

Case (i): We will show that there exists $u \in \cap_{j \in Z_t} \Omega_j$, and hence in particular u_t satisfies Ω_i. Each Ω_j can be written in the form

$$\Omega_j = \{u : \Lambda_j^{d'}(\hat{x}_{t,j}) u_t \geq \overline{\omega}_j^{d'}\} \tag{4.16}$$

where $\overline{\omega}_j^{d'}$ is a real number that does not depend on u_t. Under the assumption 1) of the theorem, there exists u satisfying (4.14) for all $i \in Z_t'$. Choose

$$u_t = \left(\max_j \{|\overline{\omega}_j^{d'}|\} / ||u||_2 \right) u.$$

This choice of u_t satisfies $u_t \in \bigcap_{j \in Z_t} \Omega_j$, in particular $u_t \in \Omega_i$.

Case (ii): In this case, Step 2 of the procedure is reached and constraints Ω_j are removed until all indices in Z_t satisfy $||\hat{x}_{t,j} - \hat{x}_{t,k}||_2 \leq \theta_{jk}$. Since $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 \leq \theta_{ij}$ already holds for all $j \in Z_t$, i will not be removed from Z_t during this step. After Step 2 is complete, the analysis of Case (i) holds and there exists a u which satisfies all the remaining constraints, including Ω_i.

Case (iii): Suppose j satisfies $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 > \theta_{ij}$. We have

$$\begin{aligned} \theta_{ij} &< ||\hat{x}_{t,i} - \hat{x}_{t,i,j} + \hat{x}_{t,i,j} - \hat{x}_{t,j}||_2 \\ &\leq ||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 + ||\hat{x}_{t,i,j} - \hat{x}_{t,j}||_2 \qquad (4.17) \\ &\leq \theta_{ij}/2 + ||\hat{x}_{t,i,j} - \hat{x}_{t,j}||_2 \qquad (4.18) \end{aligned}$$

where Eq. (4.17) follows from the triangle inequality and (4.18) follows from the assumption that $||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 \leq \theta_{ij}/2$. Hence $||\hat{x}_{t,j} - \hat{x}_{t,i,j}||_2 > \theta_{ij}/2$ and j is removed from Z_t. By applying this argument to all such indices j, we have that i is not removed during Step 2 of the procedure, and thus the analyses of Cases (i) and (ii) imply that $u_t \in \Omega_i$.

From these cases, we have that Ω_i holds whenever $\hat{h}_i^d(\hat{x}_{t,i}) \geq 0$. Therefore, by Proposition 4.1, we have

$$Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T|\ ||\hat{x}_{t,i} - \hat{x}_t||_2 \leq \gamma_i, \\ ||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 \leq \theta_{ij}/2\ \forall t) \geq \prod_{d=0}^{d'} (\frac{\hat{h}^d(x_0)}{\zeta^d}) e^{-\zeta^d T}$$

and $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T) \geq (1-\epsilon)\prod_{d=0}^{d'}(\frac{\hat{h}^d(x_0)}{\zeta^d})e^{-\zeta^d T}$ by (4.15). □

The bank of functions in Proposition 4.4 ensures the safety of the system with faulty components. Hence we define the functions as follows.

Definition 4.6. *The bank of functions $h_1^{d'}, \ldots, h_m^{d'}$ are* Fault-Tolerant High Order Stochastic Control Barrier Functions (FT-HOSCBFs) *of relative degree d' for system* (4.1)-(4.2) *if conditions in Theorem 4.4 are satisfied.*

4.3.4 Feasibility Verification

In order for Theorem 4.4 to guarantee system safety, the linear constraint (4.14) must hold for all time t. In what follows, we develop an SOS-based scheme to verify the feasibility of SCBF, FT-SCBF and FT-HOCBF constraints for both fault-free case and the case with sensor faults and attacks.

We focus on verification for a constant-gain Kalman filter. In the case where the system is LTI with constant noise, the steady-state Kalman filter gain is optimal and hence satisfies the stochastic stability criteria by Theorem 4.1. In this subsection, we omit the time subscript of x_t, $\hat{x}_t$ and z_t, i.e., (x, $\hat{x}$ and z) to simplify the expression. We consider an LTI system described by (4.1) and (4.2), where $f(x) = F$, $g(x) = G$, the matrices $R_t = R$ and $Q_t = Q$. For an LTI system, P_t is the covariance matrix for the estimation error and will converge to a steady-state value P. The Kalman filter has a constant gain given by $K = Pc^T R^{-1}$. We introduce an SOS-based approach to verify the feasibility for this case.

Verification for SCBF

We first present the verification for an SCBF in an attack-free scenario, in which one SCBF-based safety constraint must be satisfied. We have the following initial result.

Proposition 4.2. *Suppose Assumption 4.1 holds. The function $h(\hat{x})$ is a SCBF if and only if there is no $\hat{x} \in \mathcal{C}_\gamma$, $z \in \mathbb{R}^n$ satisfying $\frac{\partial h}{\partial x} g(\hat{x}) = 0$, $z^T z - \gamma^2 \leq 0$ and $\xi(\hat{x}) < 0$ where*

$$\xi(\hat{x}) = \frac{\partial h}{\partial x} f(\hat{x}) + \frac{1}{2}\mathbf{tr}\left(\nu^T K^T \frac{\partial^2 h}{\partial x^2}(\hat{x}) K\nu\right) - \left\|\frac{\partial h}{\partial x}(\hat{x}) Kc\right\|_2 \gamma + \hat{h}(\hat{x}). \tag{4.19}$$

Proof. By Theorem 4.2, the set $\mathcal{C}_\gamma$ is positive invariant given $\|x - \hat{x}\| \leq \gamma$ if for all time t u_t is chosen to satisfy (4.5) $\forall z_t$ with $\|z_t\| \leq \gamma$. By Definition 4.3, we have that $h(\hat{x})$ is a SCBF if and only if (4.5) holds for all $\hat{x} \in \mathcal{C}_\gamma := \{x : \hat{h}(x) \geq 0\}$. If $\frac{\partial h}{\partial x} g(\hat{x}) \neq 0$, we can choose u s.t.

$$\frac{\partial h}{\partial x} g(\hat{x}) u \geq \sup_{\|z\| \leq \gamma} \left\{ -\frac{\partial h}{\partial x} f(\hat{x}) - \right.$$
$$\left. \frac{1}{2} \mathbf{tr}\left(\nu^T K^T \frac{\partial^2 h}{\partial x^2}(\hat{x}) K \nu \right) + \left\| \frac{\partial h}{\partial x}(\hat{x}) K c \right\|_2 \gamma - \hat{h}(\hat{x}) \right\}.$$

Since $\|z\| \leq \gamma$ is a compact set, such a u always exists. Hence, (4) fails if and only if $\exists \hat{x}$ and z with $\|z\| \leq \gamma$ s.t. (i) $\frac{\partial h}{\partial x} g(\hat{x}) = 0$, and (ii) $\xi(\hat{x}) < 0$ hold simultaneously. □

Based on the proposition, we can formulate the following conditions via the Positivstellensatz.

Lemma 4.1. *A polynomial $h(\hat{x})$ is an SCBF for system (4.1)–(4.2) if and only if there exist polynomials $\rho(\hat{x}, z)$, sum-of-squares polynomials $q_S(\hat{x}, z)$, integers r_1 such that*

$$\phi(\hat{x}, z) + \chi(\hat{x}, z) + \psi(\hat{x}) = 0, \tag{4.20}$$

and

$$\begin{aligned} \phi(\hat{x}, z) &= \sum_{S \subseteq \{1,\ldots,3\}} q_S(\hat{x}, z) \prod_{i \in S} \phi_i(\hat{x}, z) \\ \chi(\hat{x}, z) &= (\xi(\hat{x}))^{2r_1} \\ \psi(\hat{x}) &= \sum_{i=1}^{m} \rho_i(\hat{x}, z) \left[\frac{\partial h}{\partial x} g(\hat{x}) \right]_i, \end{aligned}$$

where $\phi_1(\cdot) = -\xi(\hat{x})$, $\phi_2(\cdot) = -z^T z + \gamma^2$ and $\phi_3(\cdot) = \hat{h}(\hat{x})$.

Proof. By Proposition 4.2, we have $h(\hat{x})$ is an SCBF iff there exist no $\hat{x}$, z such that $\frac{\partial h}{\partial x} g(\hat{x}) = 0$, $z^T z - \gamma^2 \leq 0$ and $-\xi(\hat{x}) > 0$. The latter two conditions are equivalent to $-z^T z + \gamma^2 \geq 0$, and $-\xi(\hat{x}) \geq 0$, $\xi(\hat{x}) \neq 0$. These conditions are equivalent to (4.20) by the Positivstellensatz. □

Verification for FT-SCBF

We now extend the result into the case where sensors may experience faults and attacks. Specifically, we consider a nonlinear control system whose output may be affected by one of m sensor faults described by (4.7) and (4.8).

In this case, we need to verify the feasibility of u to satisfy m SCBF constraints under m possible sensor faults. To achieve this, we extend Proposition 4.2 to verify the feasibility of a set of SCBF constraints via Farkas' Lemma.

Corollary 4.1. *Define $A(x)$ and $\Xi(x)$ as follows.*

$$\begin{aligned} A(x) &= (A_1(x) \dots A_m(x))^T \\ \Xi(x) &= [\xi_1(x) \dots \xi_m(x)]^T \end{aligned} \tag{4.21}$$

Control input u that is chosen to satisfy a set of linear constraints can be written as

$$A(x)u \leq \Xi(x, z). \tag{4.22}$$

By Farkas's Lemma, the system $A(x)u \leq \Xi(x)$ has a solution $u \in \mathbb{R}^p$, if and only if there does not exist $y \in \mathbb{R}^m$ such that

$$A^T(\hat{x})y = 0,\ y \geq 0,\ \Xi^T(\hat{x})y < 0. \tag{4.23}$$

We define $A(x)$ and $\Xi(x)$ as follows

$$\begin{aligned} A(\hat{x}) &= \left(-\frac{\partial h}{\partial x} g(\hat{x}_1) \dots - \frac{\partial h}{\partial x} g(\hat{x}_m) \right)^T, \\ \Xi(\hat{x}) &= [\xi(\hat{x}_1) \dots \xi(\hat{x}_m)]^T. \end{aligned}$$

Proposition 4.3. *Suppose Assumption 4.1 and conditions in Theorem 4.4 hold. There exists a feasible solution u satisfying a set of m SCBF constraints if and only if there is no $\hat{x}_1, \dots, \hat{x}_m \in \mathcal{C}_\gamma$, $z_1, \dots, z_m \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ satisfying $z_j^T z_j - \gamma^2 \leq 0$, $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) - \gamma^2 \leq 0$, $A^T(\hat{x}_j)y = 0$, $y \geq 0$ and $\Xi^T(\hat{x}_j)y < 0$, $\forall j, k \in \{1, \dots, m\}$.*

Proof. By Definition 4.3, we have $\hat{h}(\hat{x}_j) = h(\hat{x}_j) - \overline{h}_\gamma \geq 0$ for all $\hat{x}_j \in \mathcal{C}_\gamma$. By Theorem 4.1, we have $z_j^T z_j - \gamma^2 \leq 0$ for all j. By Theorem 4.4, we have $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) \leq \gamma^2$, $\forall j, k \in \{1, \ldots, m\}$. For the case where $\hat{h}(\hat{x}) = 0$, u can be chosen to satisfy (4.5) for all j. By Corollary 4.1, we have the existence of u if and only if there does not exist $y \in \mathbb{R}^m$ such that (4.23) hold. Conversely, if for some $\hat{x}_0$, $\hat{x}_1$, z_0 and y_0 satisfying $\frac{\partial h}{\partial x} g(\hat{x}_0) = 0$, $(\hat{x}_0 - \hat{x}_1)^T(\hat{x}_0 - \hat{x}_1) - \gamma^2 \leq 0$, $z_0^T z_0 - \gamma^2 \leq 0$, $A^T(\hat{x}_0)y_0 = 0$, $y_0 \geq 0$ and $\Xi^T(\hat{x}_0)y_0 < 0$, the set $\mathcal{C}$ is not positive invariant. □

Then, we can formulate the following conditions via the Positivstellensatz.

Lemma 4.2. *There exists a feasible solution u satisfying a set of m SCBF constraints if and only if there exist polynomials $\rho(\hat{x}_j, y, z_j)$, sum-of-squares polynomials $q(\hat{x}_j, y, z_j)$, integers $s = 4m + m^2$, $r_1, \ldots, r_m$ such that*

$$\phi(\hat{x}_j, \hat{x}_k, y, z_j) + \chi(\hat{x}_j, \hat{x}_k, y, z_j) + \psi(\hat{x}_j, \hat{x}_k, y, z_j) = 0, \tag{4.24}$$

and

$$\begin{aligned}
\phi(\cdot) &= \sum_{S \subseteq \{1,\ldots,s\}} q_S(\hat{x}_j, \hat{x}_k, y, z_j) \prod_{i \in S} \phi_i(\hat{x}_j, \hat{x}_k, y, z_j) \\
\chi(\cdot) &= \prod_{\forall j \in \{1,\ldots,m\}} \left(-\Xi^T(\hat{x}_j)y\right)^{2r_j} \\
\psi(\cdot) &= \sum_{j=1}^{m} \left(\sum_{i=1}^{p} \rho_i^0(\hat{x}_j, \hat{x}_k, y, z_j) \left[A^T(\hat{x}_j)y\right]_i \right),
\end{aligned}$$

where $\phi_{\{1,\ldots,m\}}(\cdot) = -\Xi^T(\hat{x}_j)y_j$, $\phi_{\{m+1,\ldots,2m\}}(\cdot) = \hat{h}(\hat{x}_j)$, $\phi_{\{2m+1,\ldots,3m\}}(\cdot) = -z_j^T z_j + \gamma^2$, $\phi_{\{3m+1,\ldots,4m\}}(\cdot) = y_j$ and $\phi_{\{4m+1\ldots 4m+m^2\}}(\cdot) = -(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) + \gamma^2$.

Proof. By Proposition 4.3, we have $h(\hat{x})$ is an SCBF if and only if there exist no $\hat{x}_1, \ldots, \hat{x}_m$, $z_1, \ldots, z_m$ and y satisfying $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) - \gamma^2 \leq 0$, $\forall j, k \in \{1, \ldots, m\}$, $z_j^T z_j - \gamma^2 \leq 0$ $\forall j$, $A^T(\hat{x})y_j = \mathbf{0}$, $y_j \geq 0$ and $\Xi^T(\hat{x})y_j < 0$. The conditions are equivalent to $\forall j, k \in \{1, \ldots, m\}$,

$$\begin{aligned}
&- z_j^T z_j + \gamma^2 \geq 0, \\
&- (\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) + \gamma^2 \geq 0, \\
&A^T(\hat{x}_j)y = \mathbf{0},\ y \geq 0,\ -\Xi^T(\hat{x}_j)y \geq 0, \Xi^T(\hat{x}_j)y \neq 0
\end{aligned}$$

These conditions are equivalent to (4.24) by the Positivstellensatz. □

Verification for FT-SCBF with high relative degree

We further extend the proposition 4.3 and Lemma 4.2 to verify the feasibility of a set of HOSCBF constraints.

We define $A(x)$ and $\Xi(x)$ as follows

$$\begin{aligned}
A(\hat{x}) =& \left(-\frac{\partial h^{d'}}{\partial x} g(\hat{x}_1), \ldots, -\frac{\partial h^{d'}}{\partial x} g(\hat{x}_m) \right)^T, \\
\Xi(\hat{x}) =& [\xi_1(\hat{x}_1), \ldots, \xi_m(\hat{x}_m)]^T, where \\
\xi_i(\hat{x}) =& \frac{\partial h^{d'}}{\partial x} f(\hat{x}) + \frac{1}{2} \mathbf{tr} \left(\nu^T K^T \frac{\partial^2 h_i^{d'}}{\partial x^2}(\hat{x}) K \nu \right) - \\
& \left\| \frac{\partial h_i^{d'}}{\partial x}(\hat{x}) K c \right\|_2 \gamma_i + \hat{h}_i^{d'}(\hat{x}).
\end{aligned}$$

Proposition 4.4. *Suppose Assumption 4.1 and conditions in Theorem 4.4 hold. There exists a feasible solution u satisfying a set of m HOSCBF constraints with relative degree d if and only if there is no $\hat{x}_1, \ldots, \hat{x}_m \in \mathcal{C}_\gamma$, $z_1, \ldots, z_m \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ satisfying $\hat{h}^d(\hat{x}_j) \geq 0,\ \forall j, \forall d \leq d'$, $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) - \gamma^2 \leq 0$, $\forall j, k \in \{1, \ldots, m\}$, $z_j^T z_j - \gamma^2 \leq 0\ \forall j$, $A^T(\hat{x})y = 0$, $y \geq 0$ and $\Xi^T(\hat{x})y < 0$.*

Proof. By Theorem 4.1, we have $z_j^T z_j - \gamma^2 \leq 0$ for all j with m EKFs. By the Definition 4.5, we have $\hat{h}^d(\hat{x}_j) = h^d(\hat{x}_j) - \overline{h}_\gamma \geq 0$ for all $\hat{x}_j \in \overline{\mathcal{C}}$, if and only if the following three conditions are satisfied. For all $\hat{x}_j \in \overline{\mathcal{C}}$, $\hat{h}^d(\hat{x}) \geq 0$ for all $d \leq d'$. Next, by Theorem 4.4, $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) \leq \gamma^2$, $\forall j, k \in \{1, \ldots, m\}$. Moreover, $\frac{\partial h^{d'}}{\partial x} g(\hat{x}_j) \neq 0$ and u are chosen to satisfy (4.10) for all j. By Corollary 4.1, we have a solution $u \in \mathbb{R}^p$ exists, if and only if there does not exist $y \in \mathbb{R}^m$ such that (4.23) holds. Conversely, if for some $\hat{x}_0$, $\hat{x}_1$, z_0 and y_0 satisfying $\hat{h}^d(\hat{x}_0) \geq 0$, $(\hat{x}_0 - \hat{x}_1)^T(\hat{x}_0 - \hat{x}_1) - \gamma^2 \leq 0$, $z_0^T z_0 - \gamma^2 \leq 0$, $A^T(\hat{x})y_0 = 0$, $y_0 \geq 0$ and $\Xi^T(\hat{x})y_0 < 0$, the set $\mathcal{C}$ is not positive invariant. □

Lemma 4.3. *There exists a feasible solution u satisfying a set of m HOSCBF constraints if and only if there exist polynomials $\rho(\hat{x}_j, \hat{x}_k, y, z_j)$, sum-of-squares polynomials $q_S(\hat{x}_j, \hat{x}_k, y, z_j)$,*

integers $d'_1, \ldots, d'_m$, $s = 3m + m^2 + \sum_{j=1}^{m} d'_j$, $r_1, \ldots, r_m$ *such that*

$$\phi(\hat{x}_j, \hat{x}_k, y, z_j) + \chi(\hat{x}_j, \hat{x}_k, y, z_j) + \psi(\hat{x}_j, \hat{x}_k, y, z_j) = 0, \tag{4.25}$$

and

$$\begin{aligned}
\phi(\cdot) &= \sum_{S \subseteq \{1,\ldots,s\}} q_S(\hat{x}_j, \hat{x}_k, y, z_j) \prod_{i \in S} \phi_i(\hat{x}_j, \hat{x}_k, y, z_j) \\
\chi(\cdot) &= \prod_{\forall j \in \{1,\ldots,m\}} \left(-\Xi^T(\hat{x}_j) y\right)^{2r_j} \\
\psi(\cdot) &= \sum_{j=1}^{m} \left(\sum_{i=1}^{p} \rho_i^0(\hat{x}_j, \hat{x}_k, y, z_j) \left[A^T(\hat{x}_j) y\right]_i \right)
\end{aligned}$$

where

$$\begin{aligned}
\phi_{\{1,\ldots,m\}}(\cdot) &= -\Xi^T(\hat{x}_j) y \\
\phi_{\{m+1,\ldots,2m\}}(\cdot) &= y_j \\
\phi_{\{2m+1,\ldots,3m\}}(\cdot) &= -z_j^T z_j + \gamma^2 \\
\phi_{\{3m+1\ldots 3m+m^2\}}(\cdot) &= -(\hat{x}_j - \hat{x}_k)^T (\hat{x}_j - \hat{x}_k) + \gamma^2
\end{aligned}$$

and for $d_j \in \{0, \ldots, d'_j\}$, $\phi_{\{3m+m^2+1,\ldots,3m+m^2+\sum_{j=1}^{m} d'_j\}}(\cdot) = \hat{h}^{d_j}(\hat{x}_j)$.

Proof. By proposition 4.4, we have $h^d(\hat{x})$ are HOSCBFs if and only if there exist no $\hat{x}_1, \ldots, \hat{x}_m \in \mathcal{C}_\gamma$, $z_1, \ldots, z_m \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ satisfying $\hat{h}^d(\hat{x}_j) \geq 0, \forall j, \forall d \leq d'$, $(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) - \gamma^2 \leq 0$, $\forall j, k \in \{1, \ldots, m\}$, $z_j^T z_j - \gamma^2 \leq 0, \forall j$, $A^T(\hat{x}) y = 0$, $y \geq 0$ and $\Xi^T(\hat{x}) y < 0$. The conditions are equivalent to $\forall j, k \in \{1, \ldots, m\}$,

$$\begin{aligned}
&\hat{h}^{d_j}(\hat{x}_j) \geq 0, \forall d_j \leq d'_j \\
&-(\hat{x}_j - \hat{x}_k)^T(\hat{x}_j - \hat{x}_k) + \gamma^2 \geq 0, -z_j^T z_j + \gamma^2 \geq 0, \\
&A^T(\hat{x}_j) y = \mathbf{0},\ y \geq 0,\ -\Xi^T(\hat{x}_j) y \geq 0, \Xi^T(\hat{x}_j) y \neq 0
\end{aligned}$$

These conditions are equivalent to (4.25) by the Positivstellensatz. □

4.4 Joint Safety and Stability Under Sensor Faults and Attacks

We next present a framework to ensure joint safety and stability for systems with sensor faults and attacks via CLFs and HOSCBFs. Such an approach has been widely used in fault-free scenarios.

Define the goal set $\mathcal{G} \subseteq \mathcal{C}$ by $\mathcal{G} = \{x : w(x) \geq 0\}$ for some function w for some equilibrium point $x_e \in G$, $f(x_e) = 0$ and $g(x_e) = 0$. Define $\tau(\mathcal{G})$ as the first time when x_t reaches $\mathcal{G}$.

Problem Statement: Given a goal set $\mathcal{G}$, a safe set $\mathcal{C}$ and a parameter $\epsilon \in (0, 1)$, compute a control policy that, at each time t, maps the sequence $\{y_{t'} : t' \in [0, t)\}$ to an input u_t such that, given a finite stopping time T, for any fault $r \in \{r_1, \ldots, r_m\}$, $Pr(x_t \in \mathcal{C},\ 0 \leq t \leq T) \geq (1 - \epsilon)\mathcal{T}(T)$, for some function $\mathcal{T} : \mathbb{R}^n \to (0, 1)$ and $Pr(\tau(\mathcal{G}) < \infty) > 1 - \epsilon$.

4.4.1 HOSCBF-CLF

Our approach towards through asymptotically convergence to goal set $\mathcal{G}$ is through the use of *stochastic Control Lyapunov Functions*. A function $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a stochastic CLF for the SDE (4.1) if, for each x_t, we have

$$\inf_u \left\{ \frac{\partial V}{\partial x} \overline{f}(x_t, u_t) + \frac{1}{2} \mathbf{tr} \left(\sigma^T \frac{\partial^2 V}{\partial x^2} \sigma \right) \right\} < -\rho V(x_t)^\eta \tag{4.26}$$

for some $\rho > 0$ and $0 < \eta < 1$.

The following result describes the stochastic stability of systems using CLFs with [129, Theorem 3.1] providing sufficient conditions for the following result. As a preliminary, define $\tau(z) = \inf \{t : V(x_t) \leq z\}$.

Proposition 4.5 ([129, Theorem 3.1])**.** *Suppose there exists a $\overline{V}$ such that, whenever $V(x_t) \geq \overline{V}$, we choose u_t to satisfy*

$$\frac{\partial V}{\partial x} f(x_t) + \frac{\partial V}{\partial x} g(x_t) u_t + \frac{1}{2} \mathbf{tr} \left(\sigma^T \frac{\partial^2 V}{\partial x^2} \sigma \right) < -\rho V(x_t)^\eta$$

for some $\rho > 0$ *and* $0 < \eta < 1$. *For* $x \in \mathbb{R}^n \backslash \{x | V(x) = 0\}$, $Pr(\tau(\overline{V}) < \infty | x_0 = x) = 1$.

In the case with sensor faults and attacks, we consider a system with dynamics (4.1) and an Extended Kalman Filter estimator $\hat{x}_t$. The following result is an extension of Proposition 4.5 to this case.

Lemma 4.4. *Suppose that there exist constants* $M > 0$ *and* $k \in \mathbb{N}$ *such that, for any* x *and* x', $|V(x) - V(x')| \leq M||x - x'||_2^k$. *Suppose* $Pr(||\hat{x}_t - x_t||_2 \leq \gamma \ \forall t) > 1 - \epsilon$ *and, at each time* t *when* $V(\hat{x}_t) > \overline{V}$, *we have*

$$\frac{\partial V}{\partial x}(\hat{x}_t)(f(\hat{x}_t) + g(\hat{x}_t)u_t) + \gamma||\frac{\partial V}{\partial x}(\hat{x}_t)K_t c||_2 + \frac{1}{2}\mathbf{tr}\left(\nu_t^T K_t^T \frac{\partial^2 V}{\partial x^2}(\hat{x}_t)K_t\nu_t\right) < -\rho(V(\hat{x}_t) + M\gamma^k)^\eta. \quad (4.27)$$

Then

$$Pr(\tau(\overline{V} + M\gamma^k) < \infty) > 1 - \epsilon$$

for all $x_t \in \mathbb{R}^n \backslash \{x_t | V(x_t) = 0\}$.

Proof. The dynamics of $\hat{x}_t$ are given by

$$d\hat{x}_t = \overline{f}(\hat{x}_t, u_t) + K_t c(x_t - \hat{x}_t) + K_t\nu_t dV_t.$$

If $||\hat{x}_t - x_t||_2 \leq \gamma$, the differential generator $LV(\hat{x}_t)$ satisfies

$$\begin{aligned} LV(\hat{x}_t) &= \frac{\partial V}{\partial x}(\hat{x}_t)\overline{f}(\hat{x}_t, u_t) + \frac{\partial V}{\partial x}(\hat{x}_t)K_t c(x_t - \hat{x}_t) + \frac{1}{2}\mathbf{tr}\left(\nu_t^T K_t^T \frac{\partial^2 V}{\partial x^2}(\hat{x}_t)K_t\nu_t\right) \\ &\leq \frac{\partial V}{\partial x}(\hat{x}_t)\overline{f}(\hat{x}_t, u_t) + \gamma||\frac{\partial V}{\partial x}(\hat{x}_t)K_t c||_2 + \frac{1}{2}\mathbf{tr}\left(\nu_t^T K_t^T \frac{\partial^2 V}{\partial x^2}(\hat{x}_t)K_t\nu_t\right). \end{aligned}$$

since

$$\begin{aligned} \frac{\partial V}{\partial x}(\hat{x}_t)K_t c(x_t - \hat{x}_t) &\leq ||\frac{\partial V}{\partial x}(\hat{x}_t)K_t c||_2 ||x_t - \hat{x}_t||_2 \\ &\leq \gamma||\frac{\partial V}{\partial x}(\hat{x}_t)K_t c||_2. \end{aligned}$$

Since $|V(x_t) - V(\hat{x}_t)| \leq M||x_t - \hat{x}_t||_2^k$, we have $V(x_t) \leq V(\hat{x}_t) + M||x_t - \hat{x}_t||_2^k$ and $\rho(V(x_t))^\eta \leq \rho(V(\hat{x}_t) + M\gamma^k)^\eta$, for some $\rho > 0$ and $0 < \eta < 1$.

Hence, if (4.27) holds, then

$$Pr\left(\inf\{t : V(\hat{x}_t) \leq \overline{V}\} < \infty \mid ||x_t - \hat{x}_t||_2 \leq \gamma \ \forall t\right) = 1$$

by Proposition 4.5. Since $|V(x_t) - V(\hat{x}_t)| \leq M||x_t - \hat{x}_t||_2^k$, we have $V(x_t) \leq V(\hat{x}_t) + M||x_t - \hat{x}_t||_2^k$, and so

$$\begin{aligned} &Pr(V(x_t) > \overline{V} + M\gamma^k | \ ||x_t - \hat{x}_t||_2 \leq \gamma \ \forall t) \\ \leq\ &Pr(V(\hat{x}_t) + M\gamma^k > \overline{V} + M\gamma^k | \ ||x_t - \hat{x}_t||_2 \leq \gamma \ \forall t) \\ =\ &Pr(V(\hat{x}_t) > \overline{V} | \ ||x_t - \hat{x}_t||_2 \leq \gamma \ \forall t) \end{aligned}$$

Hence $Pr(\tau(\overline{V} + M\gamma^k) < \infty | \ ||x_t - \hat{x}_t||_2 \leq \gamma \ \forall t) = 1$ and thus $Pr(\tau(\overline{V} + M\gamma^k) < \infty) > 1 - \epsilon$. □

Motivated by this result, we next state a control policy that combines CLFs and HOSCBFs to ensure safety and stability. At each time t, the set of feasible control actions is defined as follows:

1. Define $Y_t(\overline{V}) = \{j : V(\hat{x}_{t,j}) > \overline{V})$, and initialize $U_t = Y_t(\overline{V})$. Define a collection of sets Υ_i, $i \in U_t$, by

$$\begin{aligned} \Upsilon_i \triangleq \Bigg\{ u : \frac{\partial V_i}{\partial x}\overline{f}(\hat{x}_{t,i}, u) + \gamma_i || \frac{\partial V_i}{\partial x}(\hat{x}_{t,i})c||_2 \\ + \frac{1}{2}\mathbf{tr}\left(\overline{\nu}_{t,i}^T K_{t,i}^T \frac{\partial^2 V}{\partial x^2}(\hat{x}_{t,i}) K_{t,i} \overline{\nu}_{t,i}\right) \\ < -\rho_i (V(\hat{x}_t) + M\gamma^k)^{\eta_i} \Bigg\} \end{aligned} \quad (4.28)$$

   for some ρ_i, η_i, $i = 1, \ldots, m$. Select any

$$u_t \in \left(\bigcap_{i \in Z_t} \Omega_i\right) \cap \left(\bigcap_{j \in U_t} \Upsilon_j\right),$$

   where Ω_i is defined as in (4.13). If no such u_t exists, go to Step 2.

2. For each i, j with $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 > \overline{\theta}_{ij}$, set $Z_t = Z_t \setminus \{i\}$ and $U_t = U_t \setminus \{i\}$ (resp. $Z_t = Z_t \setminus \{j\}$ and $U_t = U_t \setminus \{j\}$) if $||\hat{x}_{t,i} - \hat{x}_{t,i,j}||_2 > \theta_{ij}/2$ (resp. $||\hat{x}_{t,j} - \hat{x}_{t,i,j}||_2 > \theta_{ij}/2$).

If

$$\left(\bigcap_{i\in Z_t}\Omega_i\right)\cap\left(\bigcap_{j\in U_t}\Upsilon_j\right)\neq\emptyset,$$

then select u_t from this set. Else go to Step 3.

3. Remove the sets Ω_i and Υ_i corresponding to the estimators with the largest residue values until there exists a feasible u_t.

This policy is similar to the HOSCBF-based approach of Section 4.3, with additional constraints to satisfy the stability condition. This leads to another m linear inequalities. The following result gives sufficient conditions for safety and stability.

Theorem 4.5. *Suppose that $\hat{h}_1^{d'},\dots,\hat{h}_m^{d'}$, $\gamma_1,\dots,\gamma_m$, V, $\overline{V}$, and θ_{ij} satisfy the constraints of Theorem 4.4, as well as the following: (i) The function V satisfies $\{x : V(x)\leq\overline{V}+M\gamma_i^k\}\subseteq\mathcal{G}$ for all i. (ii) Define $\Gamma_i(\hat{x}_{t,i})=\frac{\partial V_i}{\partial x}g(\hat{x}_{t,i})$. Let $X_t'\subseteq X_t^{d'}(\delta)$ and $Y_t'\subseteq Y_t(\overline{V})$ be sets satisfying $||\hat{x}_{t,i}-\hat{x}_{t,j}||_2\leq\theta_{ij}$ for all $i\in X_t'$ and $j\in Y_t'$. Then there exists u with*

$$\Lambda_i^{d'}(\hat{x}_{t,i})u>0,\quad \Gamma_j(\hat{x}_{t,j})u<0 \tag{4.29}$$

for all $i\in X_t'$ and $j\in Y_t'$. If conditions (i) and (ii) hold, then $Pr(x_t\in\mathcal{C},\ 0\leq t\leq T)\geq(1-\epsilon)\prod_{d=0}^{d'}(\frac{\hat{h}^d(x_0)}{\zeta^d})e^{-\zeta^d T}$ and $Pr(\tau(\mathcal{G})<\infty)>1-\epsilon$ for any fault pattern $r\in\{r_1,\dots,r_m\}$, where $\tau(\mathcal{G})$ is the first time when x_t reaches $\mathcal{G}$.

Proof. Suppose there exists relative degree d'. By the argument of Theorem 4.4, $i\in X_t^{d'}(\delta)$ implies that Ω_i is a constraint on u_t at time t. An analogous argument yields that Υ_i is a constraint as well. By selecting u_t satisfying (4.29) at each time t, we have that $Pr(x_t\in\mathcal{C},\ 0\leq t\leq T)\geq(1-\epsilon)\prod_{d=0}^{d'}(\frac{\hat{h}^d(x_0)}{\zeta^d})e^{-\zeta^d T}$ by Theorem 4.4 and $Pr(\tau(\overline{V}+M\gamma^k)<\infty)>1-\epsilon$ by Lemma 4.4. Hence $Pr(\tau(\mathcal{G})<\infty)>1-\epsilon$ by assumption (i) of the theorem. □

A controller that reaches a goal set defined by a function V while satisfying a safety constraint $\mathcal{C}=\{x : h(x)\geq 0\}$ can be obtained by solving the optimization problem

$$\begin{array}{ll}\text{minimize} & u_t^T R u_t\\ \text{s.t.} & \Lambda_i^{d'}(\hat{x}_{t,j})u_t\geq\overline{\omega}_j^{d'}\ \forall j\in X_t^{d'}(\delta)\quad\text{(HOSCBF)}\\ & \Gamma_i(\hat{x}_{t,i})u_t\leq\overline{\tau}_i\ \forall i\in Y_t(\overline{V})\quad\text{(CLF)}\end{array} \tag{4.30}$$

at each time step, where R is a positive definite matrix representing the cost of exerting control.

4.4.2 HOSCBF-CLF Construction

As in the case of a single HOSCBF constraint, satisfaction of (4.29) will depend on the geometry of the safe region and goal set as well as the values of γ_i and θ_{ij}. We consider a linear system with dynamics

$$dx_t = (Fx_t + Gu_t)\ dt + \sigma dW_t. \tag{4.31}$$

The goal set is ellipsoidal, so that $w(x) = V(x) = (x - x'')^T \Psi (x - x'')$, and the safe region $\mathcal{C}$ is given by a hyperplane constraint $a^T x - b \geq 0$. We next construct SCBF-CLF as a special case of HOSCBF-CLF to ensure safety and stability of the cases where $\text{rank}(G) = n$ and $\text{rank}(G) < n$.

Proposition 4.6. *Suppose that rank$(G) = n$ and the following conditions hold:*

$$a^T x'' - b > 0 \tag{4.32}$$

$$\left\{ (x - x'')^T \Psi (x - x'') \leq \frac{\overline{\theta}^2 \overline{\lambda}}{2} \right\} \cap \{a^T x - b \leq 0\} = \emptyset \tag{4.33}$$

Then there exists $\delta > 0$ such that, at each time t, there exists u satisfying (4.29) when $\overline{V} = \frac{\overline{\theta}^2 \overline{\lambda}(\Phi)}{2}$.

Proof. Select δ such that $\delta < a^T x''$. We consider three cases. In the first case, $X_t(\delta) \neq \emptyset$ and $Y_t(\overline{V}) = \emptyset$. In the second case, $X_t(\delta) = \emptyset$ and $Y_t(\overline{V}) \neq \emptyset$. In the third case, $X_t(\delta) \neq \emptyset$ and $Y_t(\overline{V}) \neq \emptyset$.

If $X_t(\delta) \neq \emptyset$ and $Y_t(\overline{V}) = \emptyset$, then u satisfying $a^T G u > 0$ suffices to ensure safety by Lemma 1 in [56]. If $Y_t(\overline{V}) \neq \emptyset$ and $X_t(\delta) = \emptyset$, then choose u such that $Gu = -(\hat{x}_{t,i} - x'')$ for some $i \in Y_t(\overline{V})$. By Proposition 1 in [56], for any positive definite matrix Φ and $x' \in \mathbb{R}^n$ if

$$||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 \leq \overline{\theta} \leq \overline{\lambda}(\Phi)^{-1/2} \sqrt{2}$$

and $\hat{x}_{t,i}$ and $\hat{x}_{t,j}$ both satisfy $(x - x')^T \Phi (x - x') > (1 - \epsilon)$ for ϵ sufficiently small, then

$$(\hat{x}_{t,i} - x')^T \Phi (\hat{x}_{t,j} - x') > 0.$$

Choosing $\Phi = \frac{1}{2\overline{\theta}^2} \Psi$ and $x' = x''$ yields $\overline{\lambda}(\Phi) = \frac{1}{2\overline{\theta}^2}$, and hence $\overline{\theta} \leq \overline{\lambda}(\Phi)^{-1/2} \sqrt{2}$ holds by construction. Hence we have

$$(\hat{x}_{t,i} - x'')^T \left(\frac{1}{2\overline{\theta}^2 \overline{\lambda}} \Psi \right) (\hat{x}_{t,j} - x'') > 0$$

when $\hat{x}_{t,i}$ and $\hat{x}_{t,j}$ satisfy

$$(x - x'')^T \left(\frac{1}{2\overline{\theta}^2 \overline{\lambda}} \Psi \right) (x - x'') \geq 1,$$

or equivalently, when they satisfy

$$(x - x'')^T \Psi (x - x'') \geq \frac{\overline{\theta}^2 \overline{\lambda}}{2}.$$

Finally, suppose that $Y_t(\overline{V}) \neq \emptyset$ and $X_t(\delta) \neq \emptyset$. Choosing u such that $Gu = -(\hat{x}_{t,i} - x'')$ for some $i \in Y_t(\overline{V})$. By the preceding discussion, (4.29) holds for all $j \in Y_t(\overline{V})$. By choice of δ, $i \in X_t(\delta)$ implies that $a^T \hat{x}_{t,i} < a^T x''$. Hence, we have

$$a^T G u = a^T (x'' - \hat{x}_{t,i}) > 0,$$

and therefore $\Lambda(\hat{x}_{t,i}) u_t < 0$ is satisfied for all $i \in X_t'$. □

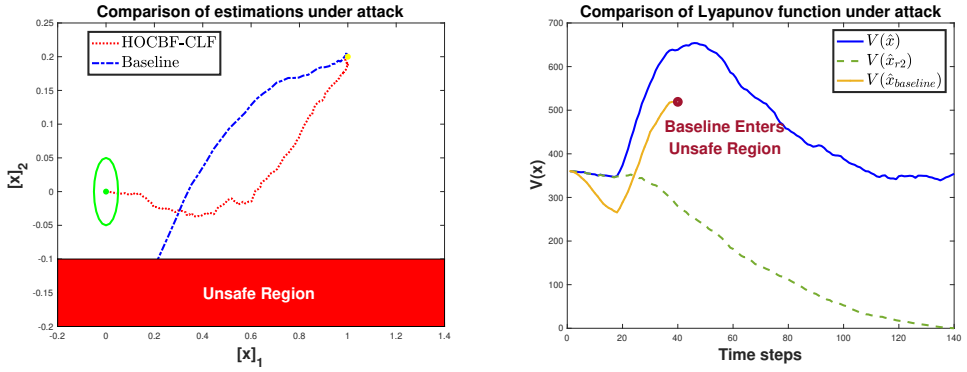
We next turn to the case where $\text{rank}(G) < n$. As in the case of the SCBF construction in [56], we add a hyperplane constraint $(x - x'')^T \Psi v < 0$ to ensure that the SCBF-CLF constraints are satisfied.

Proposition 4.7. *Suppose that $v \in \text{span}(G)$ and satisfies the following conditions: (i) $a^T v > 0$, and (ii) the initial state x' satisfies $(x' - x'')^T \Psi v < 0$. Then there exists u satisfying (4.29) at each time t.*

Proof. At each time t, choose $Gu = v$. We need to verify both SCBF constraints and the CLF constraint for each i. First, for the constraint $h(x) = a^T x - b > 0$, we must have $-a^T G u < 0$,

which is equivalent to assumption (i) of the proposition. For the constraint $(x-x'')^T\Psi Gu < 0$, the choice of $v = Gu$ and the hyperplane constraint $(x - x'')^T\Psi v < 0$ implies that the CLF constraint is satisfied. Finally, the hyperplane constraint $(x - x'')^T\Psi v < 0$ can be satisfied if $-v^T\Psi Gu < 0$, or equivalently, if $-v^T\Psi v < 0$, which holds since Ψ is positive definite. □

4.4.3 FT-CBF Evaluation



(a) (b)

Figure 4.2: Comparison of actual trajectory and Lyapunov function between HOSCBF-CLF and baseline on WMR system under sensor false data injection attacks. In (a), the baseline entered unsafe region while proposed method remains safe and converge to goal region. In (b), the Lyapunov function of real states decreases and converges to zero.

In this section, we present a case study of a wheeled mobile robot under sensor faults. We first describe the system models then then present the results.

We consider a wheeled mobile robot (WMR) with dynamics

$$\begin{pmatrix} [\dot{x}_t]_1 \\ [\dot{x}_t]_2 \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} \cos\theta_t & 0 \\ \sin\theta_t & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} [\omega_t]_1 \\ [\omega_t]_2 \end{pmatrix} + \mathbf{w}_t \tag{4.34}$$

where $([x_t]_1, [x_t]_2, \theta_t)^T$ is the vector of the horizontal, vertical, and orientation coordinates for the wheeled mobile robot, $([\omega_t]_1, [\omega_t]_2)^T$ (the linear velocity of the robot and the angular velocity around the vertical axis) is taken as the control input, and $\mathbf{w}_t$ is the process noise. The feedback linearization [130] is utilized to transform the original state vector and the WMR model into the new state variable $x_t = ([x_t]_1, [x_t]_2, [\dot{x}_t]_1, [\dot{x}_t]_2)^T$ with control input

$u_t = ([u_t]_1, [u_t]_2)^T$ and the controllable linearized model defined as follow.

$$\dot{x}_t = Fx_t + Gu_t + \mathbf{w}'_t \tag{4.35}$$

where the process noise $\mathbf{w}'_t \in \mathbb{R}^4$ has distribution $\mathcal{N}(0, \sigma_w I)$, where $\sigma_w = 0.05$. The matrices F and G are defined as

$$F = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \; G = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The following compensator is used to calculate the input $[\omega_t]_1$ and $[\omega_t]_2$ into (4.34)

$$[\omega_t]_1 = \int_{t^-}^{t^+} [u_t]_1 \cos\theta_t + [u_t]_2 \sin\theta_t \; dt \tag{4.36}$$

$$[\omega_t]_2 = ([u_t]_2 \cos\theta_t - [u_t]_1 \sin\theta_t)/[\omega_t]_1. \tag{4.37}$$

Here we assume that the observation for the orientation coordinate θ_t is attack-free and noise-free, which enables feedback linearization based on the variable θ_t.

In the linearized model, we use the observation equation

$$\begin{pmatrix} [y_t]_1 \\ [y_t]_2 \\ [y_t]_3 \\ [y_t]_4 \\ [y_t]_5 \\ [y_t]_6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} [x_t]_1 \\ [x_t]_2 \\ [\dot{x}_t]_1 \\ [\dot{x}_t]_2 \end{pmatrix} + \mathbf{a}_t + \mathbf{v}_t \tag{4.38}$$

where the measurement noise $\mathbf{v}_t \in \mathbb{R}^6$ has distribution $\mathcal{N}(0, \sigma_v I)$, where $\sigma_v = 0.05$. The impact of the attack is denoted as $\mathbf{a}_t$. The attack signal satisfies that

$$\mathbf{a}_t = \begin{cases} \mathbf{0}, & t < 1 \\ [0, 0, 0, 2, 0, 0]^T, & t \geq 1 \end{cases}$$

Note that there is one redundant sensor for the horizontal coordinate and one for the vertical coordinate.

Here we let the safe region $\mathcal{C} = \{x_t : h(x_t) = [x_t]_2 + 0.1 \geq 0, t \geq 0\}$ and the goal region $\mathcal{G} = \{x_t : \omega(x_t) = d - ||x_t - x_g||_2 \geq 0\}$, where x_g is $(0, 0)$ and $d = 0.05$ is the radius of the goal region. The baseline utilizes a fault detection scheme [112, Chapter 7.3] to detect and identify sensor faults by comparing EKF residuals against the threshold 0.1 and recomputes control input with an LQR controller. We then compare with our proposed HOCBF-based and CLF-based method. To keep the system remaining within safe region, we systematically construct the FT-SCBF with relative degree 1 by using the following class of sets

$$C^{(0)} = \{x \mid h^0(x_t) = a^T x_t + b \geq 0, \forall t \geq 0\}$$
$$C^{(1)} = \{x \mid h^1(x_t) = a^T F x_t + a^T x_t + b \geq 0, \forall t \geq 0\},$$

where $a^T = [0, 1, 0, 0]$ and $b^T = [0, 0.1, 0, 0]$. This differs from our previous work [56] which solves the problem by manually tuning the parameters and constructing the CBFs for high relative degree. In order to reach the goal region without violating the safety constraint, we choose the CLF

$$V(x) = (x_t - x_g)^T P_d (x_t - x_g) \tag{4.39}$$

where $P_d = \begin{pmatrix} \frac{1}{d} I & 0 \\ 0 & I \end{pmatrix} P_L \begin{pmatrix} \frac{1}{d} I & 0 \\ 0 & I \end{pmatrix}$, P_L is the solution of the Lyapunov equation $F^T P_L + P_L F = -I$, and I is the identity matrix [131], [132]. We set $\rho = 0.2$, $\eta = 0.8$ and $M = 2$ in the CLF constraint. The control input u_t is computed at each time step by solving (4.30) with $R = I$.

Simulation Result: The results are shown in Fig. 4.2. In Fig. 4.2(a), we plot the first two dimensions of the state, which describe the horizontal and vertical coordinates. Note that the robot stays in the safe region and eventually reaches the goal region, and hence satisfies safety and stability. As a comparison, the baseline can identify sensor faults but still resulted in a safety violation due to the slow response time of residual-based diagnosis.

4.5 Fault Tolerant NCBF

In this section, we study the problem of safety-critical control of robotic systems under sensor faults and attacks.

We consider the presence of an adversary, who can inject an arbitrary attack signal, denoted as $a_t \in \mathbb{R}^q$, to manipulate the output at each time t. The adversary aims to force the robot leaves the safety region $\mathcal{C}$. The attack signal is constrained by $\text{supp}(a_t) \subseteq \mathcal{F}(r)$, where $r \in \{r_1, \ldots, r_m\}$ is the index of possible faults or attacks, m is the total number of possible attack patterns, and $\mathcal{F}(r) \subseteq \{1, \ldots, q\}$ denotes the set of potentially compromised sensors under attack pattern r. Hence, if attack pattern r occurs, then the outputs of any of the sensors in $\mathcal{F}(r_i)$ can be arbitrarily modified. We consider that the set of possible faults or attacks is known, but the exact attack pattern that has occurred is unknown to the controller. In this section, we assume that the system satisfies Assumption 4.2. We state the problem studied in this section as follows.

Problem 4.1. *Given a safety region $\mathcal{C}$ defined in Eq. (4.3) and a parameter $\epsilon \in (0, 1)$, construct a control policy μ such that, for any attack pattern $r \in \{r_1, \ldots, r_m\}$, the probability $Pr(x_t \in \mathcal{C}\ \forall t) \geq (1 - \epsilon)$ when attack pattern r occurs.*

In what follows, we first present an overview of our solution to safety-critical control synthesis for the robot in Eq. (4.1)-(4.2) such that safety can be guaranteed under sensor faults and attacks. The key to our approach is the development of a new class of control barrier functions named *fault tolerant neural control barrier functions (FT-NCBFs)*.

4.5.1 Overview of Proposed Solution

This subsection presents our proposed solution approach to safety-critical control synthesis. Since the attack pattern is unknown, we maintain a set of m EKFs, where each EKF uses measurements from $\{1, \ldots, q\} \setminus \mathcal{F}(r_i)$ for each $i \in \{1, \ldots, m\}$. We denote the state estimates and Kalman gain obtained using $\{1, \ldots, q\} \setminus \mathcal{F}(r_i)$ as $\hat{x}_{t,i}$ and $K_{t,i}$, respectively. If there exists a function b_θ parameterized by θ such that $\mathcal{D}_\theta = \{\hat{x} | b_\theta(\hat{x}) \geq 0\} \subseteq \mathcal{C}$, then Theorem 4.2 indicates that any control input u within the feasible region

$$\Omega_i = \{u : \frac{\partial b_\theta}{\partial x} f(\hat{x}_{t,i}) + \frac{\partial b_\theta}{\partial x} g(\hat{x}_{t,i}) u - \gamma_i \| \frac{\partial b_\theta}{\partial x}(\hat{x}) K_{t,i} c_i \|_2 + \frac{1}{2} \mathbf{tr}\left(\nu_i^T K_{t,i}^T \frac{\partial^2 b_\theta}{\partial x^2}(\hat{x}_{t,i}) K_{t,i} \nu_i\right) + \hat{b}_\theta^{\gamma_i}(\hat{x}_{t,i}) \geq 0\},$$

guarantees safety under attack pattern r_i, where c_i is obtained by removing rows corresponding to $\mathcal{F}(r_i)$ from matrix c, $\hat{b}_\theta^{\gamma_i}(\hat{x}) = b_\theta(\hat{x}) - \overline{b}_\theta^{\gamma_i}(\hat{x})$, and

$$\overline{b}_\theta^{\gamma_i} = \sup_{\hat{x},\hat{x}^0} \left\{ b_\theta(\hat{x}) : ||\hat{x} - \hat{x}^0||_2 \leq \gamma_i \text{ and } b_\theta(\hat{x}^0) = 0 \right\}.$$

If there exists a control input $u \in \cap_{i=1}^m \Omega_i \neq \emptyset$, such a control input can guarantee the safety under any attack pattern r_i.

We note that the existence of a control input u satisfying the constraints specified by $\Omega_1, \ldots, \Omega_m$ simultaneously may not be guaranteed because sensor faults and attacks can significantly bias the state estimates. Thus we develop a mechanism to identify constraints conflicting with each other, and resolve such conflicts. Our idea is to additionally maintain $\binom{m}{2}$ EKFs, where each EKF computes state estimates using sensors from $\{1, \ldots, q\} \backslash (\mathcal{F}(r_i) \cup \mathcal{F}(r_j))$ for all $i \neq j$. We use a variable Z_t to keep track of the attack patterns that will not raise conflicts. The variable Z_t is initialized as $\{1, \ldots, m\}$. If $\cap_{i \in Z_t} \Omega_i = \emptyset$, we compare state estimates $\hat{x}_{t,i}$ with $\hat{x}_{t,j}$ for all $i, j \in Z_t$ and $i \neq j$. If $\|\hat{x}_{t,i} - \hat{x}_{t,j}\|_2 \geq \alpha_{ij}$ for some chosen parameter $\alpha_{ij} > 0$, then Z_t is updated as

$$Z_t = \begin{cases} Z_t \setminus \{i\}, & \text{if } \|\hat{x}_{t,i} - \hat{x}_{t,i,j}\|_2 \geq \alpha_{ij}/2 \\ Z_t \setminus \{j\}, & \text{if } \|\hat{x}_{t,j} - \hat{x}_{t,i,j}\|_2 \geq \alpha_{ij}/2 \end{cases}.$$

After updating Z_t, if $\cap_{i \in Z_t} \Omega_i \neq \emptyset$, then control input u_t can be chosen as

$$\min_{u_t \in \cap_{i \in Z_t} \Omega_i} u_t^T u_t. \tag{4.40}$$

Otherwise, we will remove indices i corresponding to attack pattern r_i causing largest residue $y_{t,i} - c_i \hat{x}_{t,i}$ until $\cap_{i \in Z_t} \Omega_i \neq \emptyset$. Here $y_{t,i}$ is the output from sensors in $\{1, \ldots, q\} \setminus \mathcal{F}(r_i)$.

The positive invariance of set $\mathcal{D}_\theta$ using the procedure described above is established in the following theorem.

Theorem 4.1 ([56]). *Suppose $\gamma_1, \ldots, \gamma_m$, and α_{ij} for $i < j$ are chosen such that the following conditions are satisfied:*

1. *Define* $\Lambda_i(\hat{x}_{t,i}) = \frac{\partial b_\theta}{\partial x} g(\hat{x}_{t,i})$. *There exists* $\delta > 0$ *such that for any* $X'_t \subseteq X_t(\delta) := \{i \mid \hat{b}_\theta^{\gamma_i}(\hat{x}_{t,i}) < \delta\}$ *satisfying* $||\hat{x}_{t,i} - \hat{x}_{t,j}||_2 \leq \alpha_{ij}$ *for all* $i, j \in X'_t$, *there exists* u *such that*

$$\Lambda_i(\hat{x}_{t,i})u > -\frac{\partial b_\theta}{\partial x} f(\hat{x}_{t,i}) + \gamma_i \| \frac{\partial b_\theta}{\partial x}(\hat{x}_{t,i}) K_{t,i} c_i \|_2 - \frac{1}{2}\mathbf{tr}\left(\nu_i^T K_{t,i}^T \frac{\partial^2 b_\theta}{\partial x^2}(\hat{x}) K_{t,i} \nu_i\right) - \hat{b}_\theta^{\gamma_i}(\hat{x}_{t,i}) \quad (4.41)$$

   for all $i \in X'_t$.

2. *For each* i, *when* $r = r_i$,

$$Pr(\|\hat{x}_{t,i} - \hat{x}_{t,i,j}\|_2 \leq \alpha_{ij}/2 \ \forall j, \|\hat{x}_{t,i} - x_t\|_2 \leq \gamma_i \ \forall t) \geq 1 - \epsilon. \quad (4.42)$$

Then $Pr(x_t \in \mathcal{D}_\theta \ \forall t) \geq 1 - \epsilon$ *for any* $r \in \{r_1, \ldots, r_m\}$.

Based on Theorem 4.1, we note that the key to our solution approach is to find the function b_θ. We name the function b_θ as *fault tolerant neural control barrier function (FT-NCBF)*, whose definition is given as below.

Definition 4.1. *A function* b_θ *parameterized by* θ *is a fault tolerant neural control barrier function for the robot in Eq.* (4.1)-(4.2) *if it there exists a control input* u *satisfying Eq.* (4.41) *under the conditions in Theorem 4.1.*

Solving Problem 4.1 hinges on the task of synthesizing an FT-NCBF for the robot in (4.1)-(4.2), which will be our focus in the remainder of this section. Specifically, we first investigate how to synthesize NCBFs when there exists no adversary (Section 4.5.2). We then use the NCBFs as a building block, and present how to synthesize FT-NCBFs. We construct a loss function to learn FT-NCBFs in Section 4.5.3. We establish the safety guarantee of our approach in Section 4.5.4.

4.5.2 Synthesis of NCBF

In this subsection, we describe how to synthesize NCBFs. We first present the necessary and sufficient conditions for stochastic control barrier functions, among which NCBFs constitute a

special class represented by neural networks. Suppose Assumption 4.1 holds. By Proposition 4.2, the function $b(\hat{x})$ is a stochastic control barrier function if and only if there is no $\hat{x} \in \mathcal{D}^\gamma := \{\hat{x} \mid \hat{b}(\hat{x}) \geq 0\}$, satisfying $\frac{\partial b}{\partial x} g(\hat{x}) = 0$ and $\xi^\gamma(\hat{x}) < 0$, (4.19).

We note that the class of NCBFs is a special subset of stochastic control barrier functions. We denote the NCBF as $b_\theta(\hat{x})$, where θ is the parameter of the neural network representing the function.

In the following, we introduce the concept of *valid* NCBFs, and present how to synthesize them. A valid NCBF needs to satisfy the following two properties.

Definition 4.2 (Correct NCBFs)**.** *Given a safety region $\mathcal{C}$, the NCBF b_θ is correct if and only if $\mathcal{D}_\theta \subseteq \mathcal{C}$.*

The correctness property requires the NCBF b_θ to induce a set $\mathcal{D}_\theta \subseteq \mathcal{C}$. If $\mathcal{D}_\theta$ is positive invariant, then $\mathcal{C}$ is also positive invariant, ensuring the robot to be safe with respect to $\mathcal{C}$. We next give the second property of valid NCBFs.

Definition 4.3 (Feasible NCBF)**.** *The NCBF b_θ parameterized by θ is feasible if and only if $\forall \hat{x} \in \mathcal{D}_\theta^\gamma := \{\hat{x} | \hat{b}_\theta^\gamma(\hat{x}) \geq 0\}$, there exists u such that $\xi_\theta^\gamma(\hat{x}) + \frac{\partial b_\theta}{\partial x} g(\hat{x}) u \geq 0$, where*

$$\xi_\theta^\gamma(\hat{x}) = \frac{\partial b_\theta}{\partial x} f(\hat{x}) + \frac{1}{2}\mathbf{tr}\left(\nu^T K_t^T \frac{\partial^2 b_\theta}{\partial x^2}(\hat{x}) K_t \nu\right) - \gamma \| \frac{\partial b_\theta}{\partial x}(\hat{x}) K_t c \|_2 + \hat{b}_\theta^\gamma(\hat{x}). \quad (4.43)$$

The feasibility property in Definition 4.3 ensures that a control input u can always be found to satisfy the inequality (4.5), and hence can guarantee safety.

We note that there may exist infinitely many valid NCBFs. In this section, we focus on synthesizing valid NCBFs that encompass the largest possible safety region. To this end, we define an operator $Vol(\mathcal{D}_\theta)$ to represent the volume of the set $\mathcal{D}_\theta$, and synthesize a valid NCBF such that $Vol(\mathcal{D}_\theta)$ is maximized. The optimization program is given as follows

$$\max_\theta \quad Vol(\mathcal{D}_\theta) \quad (4.44)$$

$$s.t. \quad \xi_\theta^\gamma(\hat{x}) \geq 0 \quad \forall \hat{x} \in \partial \mathcal{D}_\theta^\gamma \quad (4.45)$$

$$b_\theta(\hat{x}) \leq h(\hat{x}) \quad \forall \hat{x} \in \mathcal{X} \backslash \mathcal{D}_\theta \quad (4.46)$$

where $\partial\mathcal{D}_\theta^\gamma$ represents the boundary of set $\mathcal{D}_\theta^\gamma$. Here constraints (4.45) and (4.46) require parameter θ to define feasible and correct NCBFs, respectively. Solving the constrained optimization problem is challenging. In this section, we convert the constrained optimization to an unconstrained one by constructing a loss function which penalizes violations of the constraints. We then minimize the loss function over a training dataset to learn parameters θ and thus NCBF b_θ.

We denote the training dataset as $\mathcal{T} := \{\hat{x}_1, \ldots, \hat{x}_N \mid \hat{x}_i \in \mathcal{X}, \forall i = 1, \ldots, N\}$, where N is the number of samples. The dataset $\mathcal{T}$ is generated by simulating estimates with fixed point sampling as in [16]. We first uniformly discretize the state space into cells with length vector L. Next, we uniformly sample the center of discretized cell as fixed points x_f. Then we simulate the estimates by introducing a perturbation $\rho[j]$ sampled uniformly from interval $[x_f[j] - 0.5L[j], x_f[j] + 0.5L[j]]$. Finally, we have the sampling data $\hat{x}_i = x_f + \rho \in \mathcal{T} \subseteq \mathcal{X}$.

We then formulate the following unconstrained optimization problem to search for θ

$$\min_{\theta} \quad -Vol(\mathcal{D}_\theta) + \lambda_f \mathcal{L}_f(\mathcal{T}) + \lambda_c \mathcal{L}_c(\mathcal{T}) \tag{4.47}$$

where $\mathcal{L}_f(\mathcal{T})$ is the loss penalizing the violations of constraint (4.45), $\mathcal{L}_c(\mathcal{T})$ penalizes the violations of constraint (4.46), and λ_f and λ_c are non-negative coefficients. The objective function (4.44) is approximated by the following quantity

$$Vol(\mathcal{D}_\theta) = \sum_{\hat{x} \in \mathcal{T}} -ReLU(h(\hat{x}))ReLU(-b_\theta(\hat{x})). \tag{4.48}$$

Eq. (4.48) penalizes the samples $\hat{x}$ in the safety region but not in $\mathcal{D}_\theta$, i.e., $h(\hat{x}) > 0$ and $b_\theta(\hat{x}) < 0$. The penalty of violating the feasibility property in Eq. (4.45) is defined as

$$\mathcal{L}_f(\mathcal{T}) = \sum_{\hat{x} \in \mathcal{T}} -\Delta(\hat{x}) ReLU(-\xi_\theta^\gamma(\hat{x}) - \frac{\partial b_\theta}{\partial x} g(\hat{x})u + \hat{b}_\theta^\gamma(\hat{x})),$$

where $\Delta(\hat{x})$ is an indicator function such that $\Delta(\hat{x}) := 1$ if $b_\theta(\hat{x}) = \overline{b}_\theta^\gamma$ and $\Delta(\hat{x}) := 0$ otherwise. The function Δ allows us to find and penalize sample points $\hat{x}$ satisfying $\hat{b}_\theta^\gamma(\hat{x}) = 0$ and $\xi_\theta^\gamma(\hat{x}) + \frac{\partial b_\theta}{\partial x} g(\hat{x})u < 0$. For each sample $\hat{x} \in \mathcal{T}$, the control input u in $\mathcal{L}_f$ is computed as

follows

$$\begin{aligned} \min_{u} \quad & u^T u \\ s.t. \quad & \xi_\theta^\gamma(\hat{x}) + \frac{\partial b_\theta}{\partial x} g(\hat{x}) u \geq 0 \end{aligned} \tag{4.49}$$

The loss function to penalize the violations of the correctness property in Eq. (4.46) is constructed as

$$\mathcal{L}_c(\mathcal{T}) = \sum_{\hat{x} \in \mathcal{T}} ReLU(-h(\hat{x})) ReLU(b_\theta(\hat{x})) \tag{4.50}$$

Eq. (4.50) penalizes $\hat{x}$ outside the safety region but being regarded safe, i.e., $h(\hat{x}) < 0$ and $b_\theta(\hat{x}) > 0$. When $\mathcal{L}_c(\mathcal{T})$ and $\mathcal{L}_f(\mathcal{T})$ converge to 0, constraints (4.45)-(4.46) are satisfied.

4.5.3 Synthesis of FT-NCBF

In Section 4.5.2, we presented the training of NCBFs when there exists no adversary. In this subsection, we generalize the construction of the loss function in Eq. (4.47), and present how to train a valid FT-NCBF for robotic systems in Eq. (4.1)-(4.2) under unknown attack patterns. With a slight abuse of notations, we use b_θ to denote the FT-NCBF in the remainder of this section. We define $\hat{b}_\theta^{\gamma_i}(\hat{x}) = b_\theta(\hat{x}) - \overline{b}_\theta^{\gamma_i}(\hat{x})$, where

$$\overline{b}_\theta^{\gamma_i} = \sup_{\hat{x}, \hat{x}^0} \left\{ b_\theta(\hat{x}) : ||\hat{x} - \hat{x}^0||_2 \leq \gamma_i \ \text{ and } b_\theta(\hat{x}^0) = 0 \right\}.$$

The following proposition gives the necessary and sufficient conditions for a function b_θ to be an FT-NCBF.

Proposition 4.8. *Suppose Assumption 4.1 holds. The function $b_\theta(\hat{x})$ is an FT-NCBF if and only if there is no $\hat{x}_{t,i} \in \mathcal{D}_\theta^{\gamma_i}$, satisfying $\frac{\partial b_\theta}{\partial x} g(\hat{x}_{t,i}) = 0$, $\xi_\theta^{\gamma_i}(\hat{x}_{t,i}) < 0$ for all $i \in \{1, \ldots, m\}$ where*

$$\begin{aligned} \xi_\theta^{\gamma_i}(\hat{x}_{t,i}) = \frac{\partial b_\theta}{\partial x} f(\hat{x}_{t,i}) + \frac{1}{2} \mathbf{tr} \left(\nu_i^T K_{t,i}^T \frac{\partial^2 b_\theta}{\partial x^2}(\hat{x}) K_{t,i} \nu_i \right) \\ - \gamma_i \| \frac{\partial b_\theta}{\partial x}(\hat{x}_{t,i}) K_{t,i} c_i \|_2 + \hat{b}_\theta^{\gamma_i}(\hat{x}_{t,i}). \end{aligned} \tag{4.51}$$

The proposition can be proved using the similar idea to Proposition 4.2. We omit the proof due to space constraint.

We construct the loss function below to learn FT-NCBFs

$$\min_{\theta} \quad -Vol(\mathcal{D}_\theta) + \lambda_f \sum_{i\in\{1,\dots,m\}} \mathcal{L}_f^i(\mathcal{T}) + \lambda_c \mathcal{L}_c(\mathcal{T}), \tag{4.52}$$

where $\mathcal{L}_f(\mathcal{T}) = \sum_{i\in\{1,\dots,m\}} \mathcal{L}_f^i(\mathcal{T})$ is the penalty of violating the feasibility property,

$$\mathcal{L}_f^i(\mathcal{T}) = \sum_{\hat{x}\in\mathcal{T}} -\overline{\Delta}_i(\hat{x}) ReLU(-\xi_\theta^{\gamma_i}(\hat{x}) - \frac{\partial b_\theta}{\partial x} g(\hat{x})u + \hat{b}_\theta^{\gamma_i}(\hat{x})),$$

and $\overline{\Delta}(\hat{x})$ is an indicator function such that $\overline{\Delta}(\hat{x}) := 1$ if $b_\theta(\hat{x}) \leq \max_{i\in Z_t}\{\overline{b}_\theta^{\gamma_i}\}$ and $\overline{\Delta}(\hat{x}) := 0$ otherwise. The control input u used to compute $\mathcal{L}_f^i(\mathcal{T})$ for each sample $\hat{x}$ is calculated as follows.

$$\begin{aligned} \min_{u} \quad & u^T u \\ s.t. \quad & \xi_\theta^{\gamma_i}(\hat{x}) + \frac{\partial b_\theta}{\partial x} g(\hat{x})u \geq 0 \quad \forall i \in \{1,\dots,m\} \end{aligned} \tag{4.53}$$

If $\mathcal{L}_c(\mathcal{T})$ and $\mathcal{L}_f(\mathcal{T})$ converge to 0, b_θ is a valid FT-NCBF.

4.5.4 Safety Guarantee of Proposed Approach

In this subsection, we establish the safety guarantee of our approach for the robot in Eq. (4.1)-(4.2). First, we note that Theorem 4.1 establishes the positive invariance of set $\mathcal{D}_\theta$. However, the theorem depends on the existence of u_t. The following proposition provides the sufficient condition of the existence of u_t for all $\hat{x} \in \mathcal{D}_\theta^{\gamma_i},\ \forall i \in Z_t$.

Proposition 4.9. *Suppose that the interval length L used to sample the training dataset $\mathcal{T}$ satisfies $L \leq s$ and $s \to 0$. If an FT-NCBF b_θ satisfies $\mathcal{L}_f(\mathcal{T}) + \mathcal{L}_c(\mathcal{T}) = 0$, then there always exists u_t such that $\frac{\partial b_\theta}{\partial x} g(\hat{x})u_t + \xi_\theta^{\gamma_i}(\hat{x}) \geq 0\ \forall \hat{x} \in \mathcal{D}_\theta^{\gamma_i},\ \forall i \in Z_t$.*

Proof. By the constructions of $\mathcal{L}_f^i$ and $\mathcal{L}_c$, these losses are non-negative. Thus if $\mathcal{L}_f(\mathcal{T}) + \mathcal{L}_c(\mathcal{T}) = 0$, we have $\mathcal{L}_f^i(\mathcal{T}) = \mathcal{L}_f(\mathcal{T}) = \mathcal{L}_c = 0$. According to the definitions of $\mathcal{L}_f$ and $\mathcal{L}_f^i$ as well as the conditions that $L \leq s$ and $s \to 0$, we then have that there must exist some

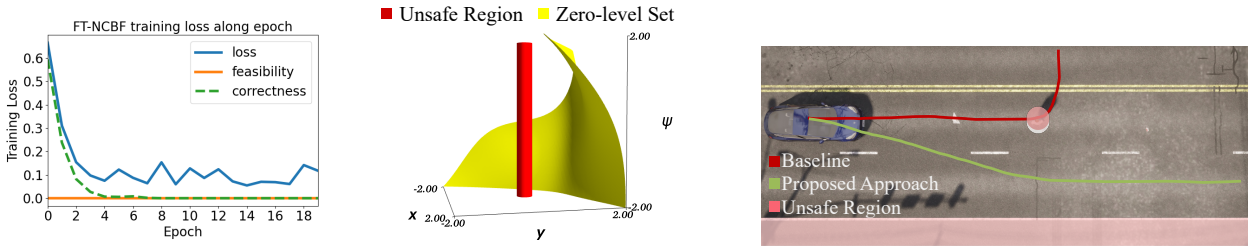
control input u that solves the optimization program in Eq. (4.53) for all $\hat{x} \in \mathcal{D}_\theta^{\gamma_i}$ when $\mathcal{L}_f^i(\mathcal{T}) + \mathcal{L}_f(\mathcal{T}) = 0$. Otherwise losses $\mathcal{L}_f^i$ and $\mathcal{L}_f$ will be positive. □

We finally present the safety guarantee of our approach.

Theorem 4.2. *Suppose that the interval length L used to sample the training dataset $\mathcal{T}$ satisfies $L \leq s$ and $s \to 0$. Let b_θ be an FT-NCBF satisfying $\mathcal{L}_f(\mathcal{T}) + \mathcal{L}_c(\mathcal{T}) = 0$. Suppose $\gamma_1, \ldots, \gamma_m$, and α_{ij} for $i < j$ are chosen such that the conditions in Theorem 4.1 hold. Then $Pr(x_t \in \mathcal{C}\ \forall t) \geq 1 - \epsilon$ for any attack pattern $r \in \{r_1, \ldots, r_m\}$.*

Proof. The theorem follows from Theorem 4.1, Proposition 4.9, and the correctness property that $\mathcal{D}_\theta \subseteq \mathcal{C}$. □

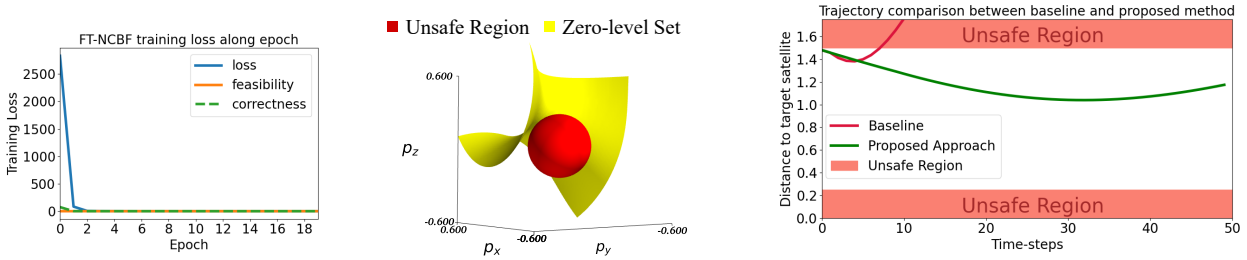
4.5.5 FT-NCBF Evaluation



(a) (b) (c)

Figure 4.3: This figure presents the experimental results on obstacle avoidance of an autonomous mobile robot. Fig. 4.3a presents the values of loss function, $\mathcal{L}_f(\mathcal{T})$, and $\mathcal{L}_c(\mathcal{T})$. The loss function decreases towards zero during the training process. Fig. 4.3b shows the zero-level set of $\mathcal{D}_\theta$ corresponding to the FT-NCBF b_θ. The set $\mathcal{D}_\theta$ does not overlap with the unsafe region in red color. Fig. 4.3c presents the trajectory of the mobile robot when using control policies obtained by our approach and the baseline approach. We observe that our approach guarantees safety whereas the baseline crashes with the pedestrian.

We evaluate our proposed approach using two case studies, namely the obstacle avoidance of an autonomous mobile robot [97] and the spacecraft rendezvous problem [99]. Both case studies are conducted on a laptop with an AMD Ryzen 5800H CPU and 32GB RAM. The hyper-parameters in both studies can be found in our code.



(a) (b) (c)

Figure 4.4: This figure presents the experimental results on spacecraft rendezvous problem. In Fig. 4.4a, we demonstrate that the value of loss function in Eq. (4.47) quickly converges to zero during training. Fig. 4.4b presents the zero-level set of $\mathcal{D}_\theta$, which never overlaps with the unsafe region in red color. Fig. 4.4c simulates the trajectories of the chaser satellite using our approach and the baseline. We observe that our approach allows the chaser satellite to maintain a proper distance to the target satellite (green curve), whereas the baseline fails (red curve).

Obstacle Avoidance Problem of Mobile Robot

We consider an autonomous mobile robot navigating on a road following the dynamics [98] given below

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

where $x := [x_1, x_2, \psi]^T \in \mathcal{X} \subseteq \mathbb{R}^3$ is the state consisting of the location (x_1, x_2) of the robot and its orientation ψ, u is the input that controls the robot's orientation, $f(x) = [\sin\psi, \cos\psi, 0]^T$, and $g(x) = [0, 0, 1]^T$.

The mobile robot is required to stay in the road while avoid pedestrians sharing the field of activities. We set the location of the pedestrian as $(0, 0)$. Then the safety region is formulated as $\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : x_1^2 + x_2^2 \geq 0.04, \text{ and } x_2 \geq -0.3\}$, where $\mathcal{X} = [-2, 2]^3$. We consider that one IMU and two GNSS sensors are mounted on the mobile robot. These sensors jointly yield the output model $y = [x_1, x_1, x_2, x_2, \psi]^T + \nu$, where the measurement noise $\nu \sim \mathcal{N}(0, \sigma I_5) \in \mathbb{R}^5$, $\sigma = 0.001$, and I_5 is the five-dimensional identity matrix. There exists an adversary who can spoof the readings from one GNSS sensor, leading to two possible attack patterns, $\{r_1, r_2\}$. The compromised sensors associated with attack patterns r_1 and r_2 are the second or fourth dimension of y, denoted as $y[2]$ and $y[4]$, respectively.

We compare our approach with a baseline which adopts the method from [16] and learns an NCBF ignoring the presence of sensor faults and attacks. The baseline computes the control input by solving $\min_{u \in \bar{\Omega}} u^T u$, where $\bar{\Omega}$ is the feasible region specified by the learned NCBF.

When applying our approach, we first sample the training dataset $\mathcal{T}$ with $L = 0.125$, making $|\mathcal{T}| = 32^3$. Given the training dataset $\mathcal{T}$, we learn an FT-NCBF using Eq. (4.52) with $\gamma_1 = 0.002$ and $\gamma_2 = 0.0015$. The training process took about 604 seconds. The values of loss function, $L_f(\mathcal{T})$, and $\mathcal{L}_c(\mathcal{T})$ at each epoch during training are presented in Fig. 4.3a. We observe that the loss function decreases towards zero during the training process. In particular, $L_f(\mathcal{T}) + \mathcal{L}_c(\mathcal{T}) \to 0$ as we train more epochs. By the construction of the loss function, it indicates that our approach finds a valid FT-NCBF. The positive invariant set $\mathcal{D}_\theta$ induced by the learned FT-NCBF is shown in Fig. 4.3b. We observe that the zero-level set $\partial\mathcal{D}_\theta$ in yellow color stays close to the boundary of the safety region, while it does not overlap with the unsafe region in red color. We implement the control policy calculated using our approach and simulate the trajectory of the mobile robot using CARLA [63]. In Fig. 4.3c, we observe that our proposed approach with parameter $\alpha_{12} = 0.1$ avoids any contact with the pedestrian while remain in the road (green color curve) and thus is safe, whereas the baseline approach (red color curve) crashes with the pedestrian and hence fails. A video clip of our simulation is available as the supplement.

Spacecraft Rendezvous Problem

We demonstrate the proposed approach using the spacecraft rendezvous between a chaser and a target satellite. We follow the setting in [99], and represent the dynamics of the satellites using the linearized Clohessy–Wiltshire–Hill equations as follows

$$\dot{x} = \begin{bmatrix} & 0_3 & & & \mathbf{I}_3 & \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} \mathbf{0}_3 \\ I_3 \end{bmatrix} u$$

where $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$ is the state of the chaser satellite, $u = [u_x, u_y, u_z]^T$ is the control input representing the chaser's acceleration, and $n = 0.056$ represents the mean-motion of the target satellite.

We define the state space and safety region as $\mathcal{X} = [-2, 2]^6$ and $\mathcal{C} = \{x : r \in [0.25, 1.5], r = \sqrt{p_x^2 + p_y^2 + p_z^2}\}$, respectively. The chaser satellite is required to maintain a safe distance from the target satellite as a safety constraint. The chaser satellite is equipped with a set of sensors to obtain the output $y = [p_x, p_x, p_y, p_y, p_y, v_x, v_y, v_z]^T + \nu$, where $\nu \sim \mathcal{N}(0, \Sigma)$ and $\Sigma = 10^{-5} \times Diag([100, 100, 100, 1, 1, 1, 1, 1])$. We consider two fault patterns $\{r_1, r_2\}$, where r_1 and r_2 are associated with compromised measurements from $y[2]$ and $y[4]$, respectively, raised by a perturbation $a \sim \mathcal{N}(-1, 0.1)$.

We evaluate our approach by comparing with the same baseline approach in Section 4.5.5. We sample from state space $\mathcal{X}$ using $L = 1$ and obtain a training dataset with $|\mathcal{T}| = 4096$. The training of FT-NCBF took about 1411 seconds with the loss $\mathcal{L}_f(\mathcal{T})$, and $\mathcal{L}_c(\mathcal{T})$ shown in Fig. 4.4a. We observe that the loss $\mathcal{L}_f(\mathcal{T})$ and $\mathcal{L}_c(\mathcal{T})$ quickly converge to 0, and thus the learned FT-NCBF is valid. We visualize the FT-NCBF b_θ in Fig. 4.4b. We synthesize a control policy using b_θ in Eq. (4.40). We observe in Fig. 4.4c that the chaser satellite never leaves the safety region using the control policy obtained by our approach, whereas the baseline fails to maintain a proper distance from the target satellite, leading to failures in the docking operation.

4.6 Conclusion

This chapter proposed a new class of SCBFs with high relative degree for safety and stability of control systems under sensor faults and attacks. Our approach maintains a set of state estimators, excludes outlier estimates and ensures safety with a CBF-based approach. We then constructed an SCBF with high order degree for each state estimator, which guaranteed safety provided that a linear constraint on the control input was satisfied at each time step. We proposed a scheme for using additional state estimators to resolve conflicts between these constraints, and derived a scheme to verify the feasibility of SCBFs. We then showed how to compose our proposed HOSCBFs with CLFs to provide joint guarantees on safety and stability of a desired goal set under sensor faults and attacks. The proposed FT-CBF against sensor faults was validated on a wheeled mobile robot. We proposed FT-NCBFs and studied the synthesis of FT-NCBFs by first deriving the necessary and sufficient conditions for FT-NCBFs to guarantee safety. We then developed a data-driven method to learn FT-NCBFs by minimizing a loss function which penalizes the violations of our derived conditions.

We demonstrated FTNCBF using the obstacle avoidance of a mobile robot and spacecraft rendezvous.

Chapter 5

Resilient Safe Control under LiDAR Perception Attacks

Autonomous Cyber-Physical Systems (CPS) fuse proprioceptive sensors such as GPS and exteroceptive sensors including Light Detection and Ranging (LiDAR) and cameras for state estimation and environmental observation. In real-world applications, system states and the environment are measured by sensors. As the environment becomes increasingly complex, modern CPS utilize exteroceptive sensors including Light Detection and Ranging (LiDAR) and cameras to obtain richer perception of the operating space [133]. Fusion among the exteroceptive sensors and proprioceptive sensors such as GPS and odometer allows CPS to better understand the environment [134] and ensure safe operation.

LiDARs, which measure the distances from the LiDAR transceiver to obstacles, provide 360° view and 3D representation, namely point cloud, of the environment rather than a 2D image as from a camera, and thus are crucial sensors for perception in autonomous vehicles (AVs). However, LiDAR sensors have been demonstrated to be vulnerable to spoofing attacks in [59]. Machine learning-based LiDAR detection was also shown to be vulnerable in [135]. LiDAR spoofing attacks focus on falsifying non-existing obstacles or hiding existing obstacles. Spoofing attacks that aim to create non-existing obstacles mainly use relay attack [57], in which an adversary fires laser to the LiDAR measurement unit with the same wavelength to inject false points. To hide an object from being detected by LiDAR sensor, methods include adversarial objects [60] and physical removal attacks [58]. Adversarial objects are synthesized such that deep neural network based detection modules fail to detect in a certain range of distance and angle. Physical removal attacks hide arbitrary objects, such as pedestrians, by relay attacks.

Modeling and detection of sensor faults and attacks have been extensively studied [50, 51, 52]. Secure system state estimation using measurements from proprioceptive sensors has been

investigated in [53, 54]. Closed-loop safety-critical control under sensor faults and attacks has been recently studied in [55, 56]. However, these approaches are applicable to CPS using only proprioceptive sensors. When exteroceptive sensors such as LiDAR are adopted by CPS, the impact of attacks on the output of the nonlinear filters used to process LiDAR measurements are not incorporated into the aforementioned safety-critical control designs [55, 56], rendering them less effective. Safety-critical control under LiDAR spoofing attacks is an open problem. To address the problem, we make the following contributions.

- We propose a fault tolerant state estimation algorithm that is resilient to attacks against proprioceptive sensors and 2D-LiDAR measurements. Our approach reconstructs a simulated scan based on a state estimate and a precomputed map of the environment. We leverage this reconstruction to remove false sensor inputs as well as detect and remove spoofed LiDAR measurements.

- We propose a fault tolerant safe control design using control barrier certificates. We present a sum-of-squares program to compute a control barrier certificate, which verifies a given safety constraint in the presence of estimation errors due to noise and attacks. We prove bounds on the probability that our synthesized control input guarantees safety.

- We validate our proposed framework using a UAV delivery system equipped with multiple sensors including a 2D-LiDAR. We show that the UAV successfully avoids the obstacles when navigating in an urban environment using our synthesized control law, while crashes into the unsafe region using a baseline.

- We propose a safe control system for 3D-LiDAR-perception-based AVs based on the point cloud from neighboring vehicles. In the proposed system, a Fault Detection, Identification, and Isolation (FDII) module detects and classifies the attacks, and updates the unsafe region for the vehicle. A safe controller guarantees the safety of the system based on the updated unsafe region.

- We analyze the correctness of the results from the FDII module. We show that the FDII module can detect and classify attacks correctly, and output the unsafe region containing the projection of the obstacles.

- Our results are validated through CARLA, in which we show that the proposed FDII procedure correctly detects multiple attack types and reconstructs the true unsafe

region. We then show that, under our control algorithm, the vehicle reaches the given target while avoiding an obstacle.

The remainder of this chapter is organized as follows. Section 5.1 presents the related work. Section 5.2 presents safety-critical control of a 2D-LiDAR-based system under sensor faults and attacks. Section 5.3 presents a fault detection, identification, isolation approach for 3D-LiDAR-based system and conduct verifiable safe control under 3D-LiDAR spoofing attacks. Section 5.4 concludes the chapter.

5.1 Related Work

False data injection (FDI) attacks have been reported in different applications, including modern power systems [51] and unmanned aerial vehicle (UAV) [136]. To this end, modeling, mitigating, and detecting FDI have been studied in [50, 51, 54, 53, 137]. LiDAR sensors have been demonstrated to be vulnerable to spoofing attacks in [59, 135]. The authors of [138] designed attacks that are capable of injecting false points at different locations in the point cloud. In [139], a stealthy attack against a perception-based controller equipped with an anomaly detector were proposed.

The existing literature on safe control in the presence of FDI attacks mainly focuses on systems with proprioceptive sensors. In [55], a barrier certificate based approach is proposed to ensure safety and reachability under FDI attack. A fault tolerant CBF is introduced in [56] to ensure joint safety and reachability under attacks targeting proprioceptive sensors. In [140], the authors have demonstrated that camera and LiDAR fusion is secure against naive attacks. For systems under attacks targeting both proprioceptive and exteroceptive sensors, how to synthesize a safety-critical control has been less studied.

LiDAR sensors have been demonstrated to be vulnerable to spoofing attacks in [59]. Machine learning-based LiDAR detection was also shown to be vulnerable in [135]. LiDAR spoofing attacks focus on falsifying non-existing obstacles or hiding existing obstacles. Spoofing attacks that aim to create non-existing obstacles mainly use relay attack [57], in which an adversary fires laser to the LiDAR measurement unit with the same wavelength to inject false points. To hide an object from being detected by LiDAR sensor, methods include adversarial objects [60] and physical removal attacks [58]. Adversarial objects are synthesized such that deep neural

network based detection modules fail to detect in a certain range of distance and angle. Physical removal attacks hide arbitrary objects, such as pedestrians, by relay attacks.

Countermeasures to LiDAR spoofing have been proposed in recent years. Defense approaches such as random sampling and randomizing waveforms focus on robust perception in a single sensor scenario. Random sampling proposed in [141] uses robust RANSAC method to randomly sample features from the point clouds. The approach presented in [142] randomizes the pulses' waveforms. However, a shortcoming in practical perception of these approaches is the increase in cost. RANSAC requires high computational capability to formulate the momentum model for adversarial detection[141]. The approach presented in [142] introduces extra modulation components into the lens system and may decrease the sensitivity of LiDAR[143]. A vehicle system is usually equipped with more than one sensors to estimate states and observe its surroundings. One cost-efficient method to increase robustness of estimation in faulty and adversarial environments is to use redundant information. Such a redundancy-based approach includes sensor fusion [144] and multiple sensor overlapping. However, existing work on fault-tolerant estimation with multiple sensor overlapping focus more on the case where an agent is equipped with redundant sensors, but leave the problem of cooperative robust perception less studied.

5.2 Resilient Safe Control of 2D-LiDAR-based Systems

We study the problem of safety-critical control of a LiDAR-based system under sensor faults and attacks. We propose a framework consisting of fault tolerant estimation and fault tolerant control. The former reconstructs a LiDAR scan with state estimations, and excludes the possible faulty estimations that are not aligned with LiDAR measurements. We also verify the correctness of LiDAR scans by comparing them with the reconstructed ones and removing the possibly compromised sector in the scan. Fault tolerant control computes a control signal with the remaining estimations at each time step. We prove that the synthesized control input guarantees system safety using control barrier certificates. We validate our proposed framework using a UAV delivery system in an urban environment. We show that our proposed approach guarantees safety for the UAV whereas a baseline fails.

5.2.1 Preliminaries

Consider a discrete-time control-affine system given as:

$$x[k+1] = f(x[k]) + g(x[k])u[k] + w[k] \tag{5.1}$$

where $w[k]$ is a Gaussian process with mean zero and autocorrelation function $R_w(k, k') = Q_k\delta(k - k')$ with δ denoting the discrete-time delta function and Q_k is a positive definite matrix. We assume that there is a nominal controller $u = \pi(x)$, for some function $\pi : \mathcal{X} \to \mathbb{R}^m$. We let $x[k] \in \mathcal{X} \subseteq \mathbb{R}^n$ denote the system state and $u[k] \in \mathbb{R}^m$ denote a control signal at time k. Functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be Lipschitz continuous.

System (5.1) uses a set of sensors $I_p := \{1, \ldots, n_p\}$ to measure its states with observation $y[k] \in \mathbb{R}^z$ following the dynamics described as:

$$y[k] = o(x[k]) + v[k], \tag{5.2}$$

where $o : \mathbb{R}^n \to \mathbb{R}^z$ is the observation function, $v[k]$ is an independent Gaussian process with mean identically zero and autocorrelation function $R_v[k, k'] = R_k\delta(k - k')$ and R_k is a positive definite matrix. In what follows, we give background on discrete-time Extended Kalman Filter (EKF) and estimating pose from LiDAR scans

DT-EKF

For the system with dynamics (5.1) and observation (5.2), the state estimate $\hat{x}$ is computed via EKF as:

$$\hat{x}[k+1] = F(\hat{x}[k], u[k]) + K_k(y[k] - o(\hat{x}[k])), \tag{5.3}$$

where $F(x[k], u[k]) = f(x[k]) + g(x[k])u[k]$. The Kalman filter gain is

$$K_k = A_k P_k C_k^T (C_k P_k C_k^T + R_k)^{-1}, \tag{5.4}$$

where $A_k = \frac{\partial F}{\partial x}(\hat{x}[k], u[k])$, $C_k = \frac{\partial o}{\partial x}(\hat{x}[k])$, and P_k is defined by the Riccati difference equation:

$$P_{k+1} = A_k P_k A_k^T + Q_k - K_k(C_k P_k C_k^T + R_k)K_k^T.$$

The error bound of discrete-time EKF can be derived by Theorem 3.2 in [145] if Assumption 5.1 holds.

Assumption 5.1. *The system described by* (5.1) *and* (5.2) *satisfies the conditions:*

- A_k *is nonsingular for every* $k \geq 0$.
- *There are positive real numbers* $\bar{a}, \bar{c}, \underline{p}, \bar{p} > 0$ *such that the following bounds on various matrices are fulfilled for every* $k \geq 0$ *:*

$$\|A_k\| \leq \bar{a};\ \|C_k\| \leq \bar{c};\ \underline{p}I \leq P_k \leq \bar{p}I;$$
$$\underline{q}I \leq Q_k;\ \underline{r}I \leq R_k.$$

- *Let* ϕ *and* χ *be defined as*

$$\begin{aligned} F(x[k], u[k]) - F(\hat{x}[k], u[k]) &= A_k(x[k] - \hat{x}[k]) \\ &\quad + \varphi(x[k], \hat{x}[k], u[k]) \\ o(x[k]) - o(\hat{x}[k]) &= C_k(x[k] - \hat{x}[k]) + \chi(x[k], \hat{x}[k]) \end{aligned}$$

  Then there are positive real numbers $\epsilon_\varphi, \epsilon_\chi, \kappa_\varphi, \kappa_\chi > 0$ *such that the nonlinear functions* φ, χ *are bounded via*

$$\|\varphi(x, \hat{x}, u)\| \leq \kappa_\varphi \|x - \hat{x}\|^2, \quad \|\chi(x, \hat{x})\| \leq \kappa_\chi \|x - \hat{x}\|^2$$

  for $x, \hat{x} \in R^n$ *with* $\|x - \hat{x}\| \leq \epsilon_\varphi$ *and* $\|x - \hat{x}\| \leq \epsilon_\chi$*, respectively.*

If the conditions of Assumption 1 hold, the estimation error $\zeta_k = x[k] - \hat{x}[k]$ is exponentially bounded in mean square and bounded with probability one, provided that the initial estimation error satisfies $\|\zeta_0\| \leq \bar{\zeta}$ [145].

5.2.2 2D LiDAR Observation and Threat Model

The system is equipped with a LiDAR sensor that observes the environment by calculating the ranges and angles to objects. A LiDAR sensor fires and collects n_s laser beams to construct a scan $S := \{(s_i^r, s_i^a),\ 0 \leq i \leq n_s\}$, where s_i^r denotes the range of the i-th scan, and s_i^a denotes

the angle of the i-th scan. We denote the Cartesian translated LiDAR scan S measured at pose x as $\mathcal{O}(x, S)$.

We assume a 2D point-cloud map $\mathcal{M}$ is known by the system as prior knowledge. The map $\mathcal{M} := \{(m_i^x, m_i^y),\ 0 \leq i \leq n_{\mathcal{M}}\}$ is a collection of $n_{\mathcal{M}}$ points with tuples of object positions (m_i^x, m_i^y) in the world coordinate.

We assume that there exists an adversary that aims to cause collisions or other unsafe behaviors. The adversary has the capability to utilize any state-of-the-art spoofer for different sensors to conduct false data injection to perturb the observations. The injected false data denoted as a can bias the system state estimation and cause the system to make incorrect control decisions. We denote the perturbed observations as

$$\bar{y}[k] = o(x[k]) + v[k] + a[k]. \tag{5.5}$$

The adversary can also compromise the LiDAR sensor by creating a near obstacle as demonstrated in [138]. The adversary fires laser beams to inject several artificial points e' into a LiDAR scan. We denote the compromised LiDAR scan as $S \oplus e'$, where $\oplus$ is a merge function introduced by [138]. However, due to the physical limitation of spoofer hardware, the injected point can only be within a very narrow spoofing angle, i.e. 8° horizontal angle.

We index the LiDAR sensor as the 0-th sensor and define $I = \{0\} \bigcup I_p$. We denote the set of sensors attacked by the adversary as $\mathcal{A} \subseteq I$. We assume that the system is uniformly observable from the sensors in $I \backslash \mathcal{A}$. We assume that, at each time k, the support of $a[k]$ is contained in $\mathcal{A}$.

We define the state space $\mathcal{X}$ and a safety set $\mathcal{C}$ as

$$\mathcal{X} = \{x : h(x) \geq 0\}, \quad \mathcal{C} = \{x \in \mathcal{X} : h_0(x) \geq 0\},$$

where $h, h_0 : \mathcal{X} \mapsto \mathbb{R}$. We say system (5.1) is safe with respect to $\mathcal{C}$ if $x[k] \in \mathcal{C}$ for all time $k = 0, 1, \ldots$. We assume that the safe region $\mathcal{C}$ is pre-defined and known by the system, and the initial state of the system is safe, i.e. $x_0 \in \mathcal{C}$.

Problem 5.1. *Given a map $\mathcal{M}$ and a safety set $\mathcal{C}$, we consider a nonlinear LiDAR-based system with dynamics* (5.1) *that is controlled by a nominal controller. The problem studied*

is to find a scheme to ensure system safety with desired probability $(1-\epsilon)$*, where* $\epsilon \in (0,1)$*, when an adversary is present.*

Estimating Pose By Comparing Scans

Pose refers to the position of the system in a Cartesian coordinate frame. Pose estimations with LiDAR scans have been extensively studied. NDT [146], as one of the widely-used approaches, models the distribution of all reconstructed 2D-Points of one laser scan by a collection of local normal distributions.

Consider two states $x_1, x_2 \in \mathcal{X}$ and the LiDAR scans $\mathcal{O}(x_1, S_1)$ and $\mathcal{O}(x_2, S_2)$ collected at x_1 and x_2, respectively. The NDT method estimates the relative pose change as $r = \mathcal{O}(x_1, S_1) \ominus \mathcal{O}(x_2, S_2)$, where $\ominus$ is a scan match operation. The scan match operation is implemented as follows. The NDT method first subdivides the surrounding space uniformly into cells with constant size. For each cell in $\mathcal{O}(x_1, S_1)$, the mean q and the covariance matrix Σ are computed to model the points contained in the cell as the normal distribution $N(q, \Sigma)$. Denote the points in $\mathcal{O}(x_2, S_2)$ as $p_i,\ i \in n_s$, where p_i is a position vector and n_s is the number of valid points. Define loss function $\mathcal{L}_s(r')$ as

$$\mathcal{L}_s(r') = \sum_i \exp\left(\frac{-((p_i - r') - q_i)^T \Sigma_i^{-1}((p_i - r') - q_i)}{2}\right) \tag{5.6}$$

The relative pose change r' is estimated by solving the minimization problem

$$\min_{r'} -\mathcal{L}_s(r') \tag{5.7}$$

with Newton's algorithm. We use r to denote the solution to (5.7) for the rest of the paper. The corresponding loss $\mathcal{L}_s(r)$ can be computed with the output of scan match r by (5.6).

5.2.3 2D-LiDAR Fault Tolerant Safe Control

In this section, we propose a framework for safe control that is compatible with existing LiDAR-based autonomous systems. We first give an overview and then describe each component in detail.

Overview of Framework

We consider a system with dynamics (5.1) and observation model (5.2) in the presence of an adversary, as described in Section 5.2.1. To guarantee the system's safety under attacks, we propose a fault tolerant framework to ensure safety at each time step. The framework consists of two parts, namely *fault tolerant estimation* and *fault tolerant control*.

The idea of fault tolerant estimation is to exclude compromised sensors in I_p by utilizing additional information contained in LiDAR sensor measurements. We maintain a set of state estimations $\hat{x}_i$ using EKF, where $i \in I_l \subseteq 2^{I_p}$ and each element of $i \in I_l$ is a collection of sensors in I_p such that system (5.1) is uniformly observable from the sensors in I_l. As shown in Fig. 5.1, a fault tolerant estimation reconstructs a LiDAR observation, denoted as $\mathcal{O}(\hat{x}_i, \mathcal{M})$, for each state estimation $\hat{x}_i$. The reconstruction is achieved by simulating the scan process on knowledge map $\mathcal{M}$ with state estimate $\hat{x}$ being the center. We propose a fault tolerant LiDAR estimation to compare the estimated LiDAR scan $\mathcal{O}(\hat{x}_i, \mathcal{M})$ with the actual LiDAR measurement $\mathcal{O}(x, S)$. The comparison then provides a pose estimation. Using the pose estimation, our proposed fault tolerant state estimation excludes the conflicting state estimations, i.e., the state estimations that deviate from the LiDAR estimation.

After excluding the conflicting state estimations using fault tolerant estimation, we then design fault tolerant safe control to ensure safety of the system at each time step. Fault tolerant safe control computes an input u_o that does not deviate too far from the nominal controller $\pi(\hat{x}_i)$ for all i given by the fault tolerant estimation. The safety of u_o is certified by a discrete-time barrier certificate.

In what follows, we describe the fault tolerant estimation in two-fold, that is fault tolerant LiDAR estimation (Section 5.2.3) and fault tolerant state estimation (Section 5.2.3).

Fault Tolerant LiDAR Estimations

In the following, we introduce fault tolerant LiDAR estimation. This procedure converts each state estimation $\hat{x}_i$ given by EKFs to an estimated LiDAR observation $\mathcal{O}(\hat{x}_i, \mathcal{M})$ using map $\mathcal{M}$. The estimated LiDAR observation is then compared with the actual LiDAR observation to exclude possible faults in state estimations.

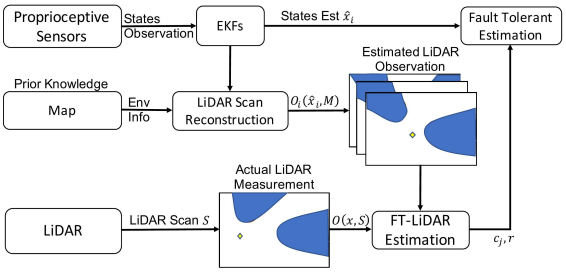


Figure 5.1: Fault tolerant estimation for LiDAR-based system removes conflicting state estimations by comparing estimations of proprioceptive sensors with additional information from exteroceptive sensors measurements.

Fault tolerant LiDAR estimation is presented in Alg. 8. Given parameters on the resolution of the LiDAR scan c_r and maximum LiDAR range r_{max}, we initialize the estimated scan $S_{\mathcal{M}} = \{(l_k^r, l_k^a)\}$ with a circle centered at $\hat{x}_i$ and radius as r_{max}. We equally divide the circle and assign sectors S_k to the corresponding l_k^a. Next, we represent the points in map $\mathcal{M}$ using polar coordinates with the origin at $\hat{x}_i$. To simulate the scan, we assign the closest point to the scan from line 6 to 10. We iterate through all points $m_j \in \mathcal{M}$. For the point in sector S_k, we replace l_k^r with m_j^r if $m_j^r \leq l_k^r$. Then we remove the points that have never been updated. Finally, we output estimated observation $\mathcal{O}(\hat{x}_i, \mathcal{M}) = \mathcal{O}(\hat{x}_i, S_{\mathcal{M}})$. Intuitively, this estimated observation can be viewed as the output of a LiDAR scan centered at state $\hat{x}_i$ with object locations given in the map $\mathcal{M}$. Hence, any deviation of the estimated and actual scans indicates either an error in the state estimate or a spoofing attack on the scan.

Next, we consider the case where the adversary not only injects false data into prorioceptive sensors but also spoofs LiDAR sensors. The intuitive countermeasure is to remove the region of the scan that is impacted by false data. Since the adversary is only capable of modifying points in the scan within a narrow spoofing angle, our approach is to partition the scan and map into regions c_j and attempt to identify which region has been impacted by spoofing. That region is then removed from the scan and the estimated scan. Since the adversary tries to bias the state estimation, we model the problem of choosing a set of observations to ignore in order to mitigate the impact of false data as a minimax optimization

$$\min_{e_I'} \max_{c_j} \mathcal{L}_s(\tilde{r}) \tag{5.8}$$

where $\tilde{r} = \mathcal{O}(\hat{x}_i, \mathcal{M} \backslash c_j) \ominus \mathcal{O}(x, S \oplus e_I' \backslash c_j)$. We search for subdivision c_j through the LiDAR observation space with Alg. 9, which is detailed as follows.

The adversary compromises the LiDAR scan S by merging it with false data e'_I, denoted as $S \oplus e'_I$. As shown in Alg. 9, we take in state estimation $\hat{x}_i$, number of sectors n_j, map $\mathcal{M}$, and scan S to search for sector c_j over scan S. The algorithm outputs the corresponding estimated relative pose $\tilde{r}_j$. For each sector c_j, we estimate observations $\mathcal{O}(\hat{x}_i, \mathcal{M}\backslash c_j)$ with Alg. 8 and reconstruct the corresponding LiDAR observation $\mathcal{O}(x, S \oplus e'_I \backslash c_j)$. Next, we compute n_s^j, the number of points contained in $S\backslash c_j$, and perform scan match to obtain $\tilde{r}$ by

$$\tilde{r}_j = \mathcal{O}(\hat{x}_i, \mathcal{M}\backslash c_j) \ominus \mathcal{O}(x, S \oplus e'_I \backslash c_j). \tag{5.9}$$

Then, we compute the loss function $\mathcal{L}_s(\tilde{r})$ and the performance degradation $\zeta_s^j = n_s^j - \mathcal{L}_s(\tilde{r})$. Finally, we output $\tilde{r}_j$ and c_j for $\zeta_s^j \leq \bar{\zeta}_s$. In what follows, we compute the upper bound $\bar{\zeta}_s$ of the degradation of the loss $\mathcal{L}_s$ brought by noise as the criteria of whether LiDAR sensor is affected by factors other than noise. We consider a point p_i sampled in the LiDAR scan collected at state x with a zero-mean disturbance w_i whose norm is bounded as $\|w_i\| \leq \bar{w}_i$.

Theorem 5.1. *Consider a state x and its state estimation $\hat{x}$. Let $\mathcal{O}(x, S)$ and $\mathcal{O}(\hat{x}, \mathcal{M})$ be LiDAR scan and estimated LiDAR observation. Let $r = \mathcal{O}(\hat{x}_i, \mathcal{M}) \ominus \mathcal{O}(x, S)$ and $\tilde{r}$ be computed by (5.9) when adversary present. In the case where the LiDAR sensor is not attacked, we have the performance degradation ζ_s is bounded by*

$$\begin{aligned} \zeta_s &:= \mathcal{L}_s^{max}(r) - \mathcal{L}_s(r) \\ &\leq n_s - \sum_i \exp\left(\frac{-\bar{w}_i^2 \lambda(\Sigma_i^{-1})}{2}\right) =: \bar{\zeta}_s, \end{aligned} \tag{5.10}$$

where $\mathcal{L}_s^{max}(r)$ is the maximum of (5.6), n_s is the number of points contained in S, and $\lambda(\Sigma_i^{-1})$ is the maximum eigenvalue of Σ_i^{-1}.

When the LiDAR sensor is attacked, if a subdivision $c_j \supseteq e'_I$ can be found by Alg. 9, we have the performance degradation of scan match is bounded as (5.10), where n_s is the number of points contained in $S\backslash c_j$ and the summation is over all points in $S \setminus c_j$.

Proof. We first show that $L_s^{max}(r) = n_s$. Then we derive a lower bound for $\mathcal{L}_s(r)$. Since covariance Σ_i is positive definite, using (5.6) we have

$$L_s^{max}(r) = \sum_i \exp\left(\frac{-((p_i - r) - q_i)^T \Sigma_i^{-1}((p_i - r) - q_i)}{2}\right) \leq \sum_i \exp(0) = n_s.$$

Let p_i be a point sampled in LiDAR scan. We have that $((p_i - r) - q_i) \leq= w_i$ with w_i being the realized disturbance when sampling p_i. Since $\|w_i\| \leq \bar{w}_i$ and Σ_i^{-1} is Hermitian, we then have

$$\sum_i \exp\left(\frac{-((p_i - r) - q_i)^T \Sigma_i^{-1}((p_i - r) - q_i)}{2}\right) \geq \sum_i \exp\left(\frac{-\bar{w}_i^2 \lambda(\Sigma_i^{-1})}{2}\right).$$

Hence, we have that ζ_s is bounded as (5.10).

When the LiDAR sensor is spoofed, there always exists a subdivision c_j such that the false data e'_I satisfies $e'_I \subseteq c_j$. If c_j is successfully identified by Alg. 9, then the subdivision c_j along with the false data e'_I are ignored. In this case, our analysis for the scenario where the LiDAR sensor is not attacked can be applied, yielding the bound in (5.10) with n_s being the number of points contained in $S \setminus c_j$. If c_j containing e'_I is not identified and is not ignored, then by line 10 of Alg. 9, we have that $\zeta_s^j \leq \bar{\zeta}_s$ and thus the bound in (5.10) follows. □

Fault Tolerant State Estimation

We next propose the criteria to develop an algorithm for a fault tolerant state estimation that provides bounded estimation error under false data attacks on the proprioceptive sensors. Our approach computes a set of indices $I_a \subseteq I_l$ that are removed to ensure that the state estimation error is bounded. Given a state estimation deviation threshold θ_h and a scan match degradation threshold $\bar{\zeta}_s$, a state estimate is *not removed* (i.e. $i \notin I_a$) if either of the following criteria holds:

- Case I: $i \notin I_a$ for estimation indexed $i \in I_l$, if $\|r_i\| \leq \theta_h$ and $\zeta_s^i \leq \bar{\zeta}_s$.
- Case II: $i \notin I_a$ for estimation indexed $i \in I_l$, if $\|\tilde{r}_i\| \leq \theta_h$ and $\tilde{\zeta}_s^i \leq \bar{\zeta}_s$.

We consider LiDAR observation is trusted, if for all $i \in I_l$ estimated LiDAR observation, the scan match degradation $\zeta_s^i \leq \bar{\zeta}_s$. In Case I, we have the scan match degradation $\zeta_s^i \leq \bar{\zeta}_s$, and the pose deviation $\|r_i\| \leq \theta_h$. We draw the conclusion that $\hat{x}_i$ agrees with the LiDAR observation, and hence $i \in I \backslash I_a$. When the LiDAR observation is not trusted, we reconstruct estimated and actual LiDAR observation with Alg. 9 to exclude section c_j. In Case II, we have the reconstructed scan match degradation $\|\tilde{r}_i\| \leq \theta_h$, and the pose deviation within tolerance with $\tilde{\zeta}_s^i \leq \bar{\zeta}^s$. We draw the conclusion that $i \in I \backslash I_a$.

In what follows, we show that sensor $i \in I \backslash I_a$ selected by criteria is attack-free and we can further have the deviation of FT-Estimation bounded by the EKF error bound of selected sensors.

Theorem 5.2. *Given scan match results r_i, $\tilde{r}_i$ and $\bar{\zeta}_s$, for sensor $i \in I \backslash I_a$ given by criteria I and II, we have estimation error bounded as $\|x - \hat{x}_i\| \leq \bar{\zeta}_i$.*

Proof. We prove by contradiction. We suppose that there exists a sensor $b \in I \backslash I_a$, whose estimation $\hat{x}_b$ satisfies $\|x - \hat{x}_b\| > \bar{\zeta}_b$. We next show contradictions for Case I and II.

In Case I, set $\theta_h = \min_i \bar{\zeta}_i$. Since $\zeta_s^b \leq \bar{\zeta}_s$, we have that LiDAR scan matches with estimated scan with relative pose change $r = x - \hat{x}_b$. If sensor b is included in $I \backslash I_a$, we have $\|x - \hat{x}_b\| \leq \theta_h \leq \bar{\zeta}_b$, which contradicts to $\|x - \hat{x}_b\| > \bar{\zeta}_b$.

In Case II, set $\theta_h = \min_i \bar{\zeta}_i$. Since $\tilde{\zeta}_s^b \leq \bar{\zeta}_s$, we have that LiDAR scan matches with estimated scan with relative pose change $\tilde{r} = x - \hat{x}_b$. If sensor b is included in $I \backslash I_a$, we have $\|x - \hat{x}_b\| \leq \theta_h \leq \bar{\zeta}_b$, which contradicts to $\|x - \hat{x}_b\| > \bar{\zeta}_b$.

Otherwise, sensor b will be excluded into set I_a and hence for any sensor $i \in I \backslash I_a$ we have the error bounded. □

5.2.4 Fault-Tolerant Control Barrier Certificate

We next present the fault tolerant control synthesis to ensure safety of the system. We set the state estimation as $\hat{x}_\alpha[k] = \hat{x}_i$, for some $i \in I \backslash I_a$. We define the control input signal as $u_o[k] = \pi(\hat{x}_\alpha[k]) + \hat{u}[k]$. In what follows, we assume the nominal controller is of the form $\pi(x) = \pi_0 + K_c\hat{x}_\alpha$ for some $\pi_0 \in \mathbb{R}^m$ and matrix K_c. Since we have $||x[k] - \hat{x}_\alpha[k]|| \leq \bar{\zeta}_\alpha$ by Theorem 5.2, the nominal control input for the estimated state satisfies

$$||\pi(\hat{x}_\alpha[k]) - \pi(x[k])|| = ||K_c(x[k] - \hat{x}_\alpha[k])|| \leq ||K_c||\bar{\zeta}_\alpha.$$

Hence, if we choose $u_o[k]$ such that $||\hat{u}[k]||_2 \leq \xi - ||K_c||\bar{\zeta}_\alpha$ for some $\xi \geq 0$, then we can guarantee that the chosen control input is within a bounded distance of the nominal control input corresponding to the true state value.

Proposition 5.1. *Consider a discrete-time system described by* (5.1) *and sets* $\mathcal{C}, \mathcal{D} \subseteq \mathcal{X}$*. If there exist a function* $B : \mathcal{X} \to \mathbb{R}_0^+$*, a constant* $c \geq 0$*, a linear controller* $u = K_c x$*, and a constant* $\gamma \in [0, 1)$ *such that*

$$B(x) \leq \gamma, \qquad \forall x \in \mathcal{C} \tag{5.11}$$

$$B(x) \geq 1, \qquad \forall x \in \mathcal{D} \tag{5.12}$$

$$\begin{aligned} \mathbb{E}[B(f(x) + g(x)(K_c x + \hat{u}) \\ + w) \mid x] \leq B(x) + c, \end{aligned} \qquad \forall x \in \mathcal{X}, \forall \|\hat{u}\| \leq \xi \tag{5.13}$$

then for any initial state $x_0 \in \mathcal{C}$*, we have the* $Pr(x[k] \in \mathcal{C}, 0 \leq k \leq T_d) \geq 1 - \gamma - cT_d$*.*

Proof. We have $u_o - u = K_c x - K_c\hat{x}_\alpha + \hat{u}$. Since $\|\hat{u}\| \leq \xi - K_c\bar{\zeta}_\alpha$ and $\|K_c x - K_c\hat{x}_\alpha\| \leq K_c\bar{\zeta}_\alpha$. By triangle inequality, we can have $\|u_o - u\| \leq \xi$. Since there exists a function $B(x)$ satisfying (5.11) to (5.13), $B(x)$ is a control barrier certificate for system (5.1). According to [17] and (5.12), we have

$$\begin{aligned} &Pr\left\{x[k] \in \mathcal{D} \text{ for some } 0 \leq k < T_d \mid x(0) = x_0\right\} \\ &\qquad \leq Pr\left\{\sup_{0 \leq k < T_d} B(x[k]) \geq 1 \mid x(0) = x_0\right\} \\ &\qquad \leq B\left(x_0\right) + cT_d \leq \gamma + cT_d. \end{aligned}$$

□

We define $h_B^{\xi}(\hat{u}) = (\xi - K_c\bar{\zeta}_\alpha)^2 - \|\hat{u}\|_2^2$. The system has continuous state space $\mathcal{X}$ and action space U, we can follow the standard procedure to compute control barrier certificate $B(x)$ by solving an SOS programming given as follows:

Proposition 5.2. *Suppose there exist a function $B(x)$ and polynomials $\lambda_0(x)$, $\lambda_1(x)$, $\lambda_x(x,\hat{u})$ and $\lambda_{\hat{u}}(x,\hat{u})$ such that*

$$-B(x) - \lambda_0(x)h_0(x) + \gamma \text{ is SOS} \tag{5.14}$$

$$B(x) + \lambda_1(x)h_0(x) - 1 \text{ is SOS} \tag{5.15}$$

$$-\mathbb{E}[B(f(x) + g(x)(K_c x + \hat{u}) + w) \mid x] + B(x) - \lambda(x)h(x) - \lambda_{\hat{u}}(x,\hat{u})h_B^{\xi}(\hat{u}) + c \text{ is SOS} \tag{5.16}$$

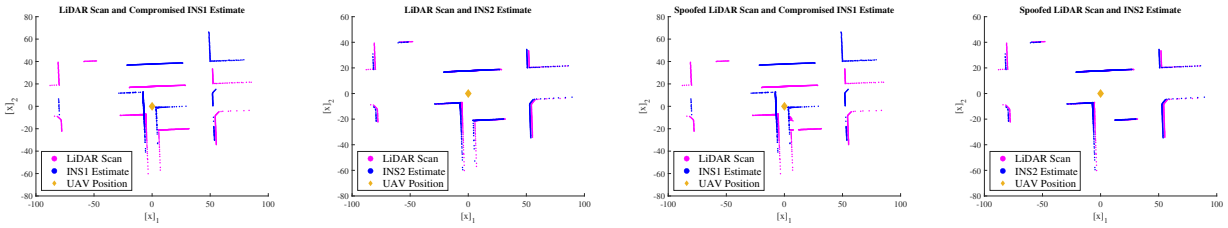
then for any initial state $x_0 \in \mathcal{C}$, we have the $Pr(x[k] \in C, 0 \le k \le T_d) \ge 1 - \gamma - cT_d$.

Proof. Since the entries $B(x)$ and $\lambda_0(x)$ in $-B(x) - \lambda_0(x)h_0(x) + \gamma$ are SOS, we have $0 \le B(x) + \lambda_0(x)h_0(x) \le \gamma$. Since the term $\lambda_0(x)h_0(x)$ is nonnegative over $\mathcal{C}$, (5.14) and (5.15) implies (5.11) and (5.12) in Proposition 5.1. Since the terms $\lambda_{\hat{u}}(x)h_B^{\xi}(\hat{u})$ and $\lambda(x)h(x)$ are nonnegative over set $\mathcal{X}$, we have (5.13) holds, which implies that the function $B(x)$ is a control barrier certificate. □

The choice of ξ uses a similar approach as [55] by solving the SOS program offline to enhance the scalability. Other numerical issues of SOS program such as sparsity and ill-conditioned problem are investigated in [147, 148].

We propose Alg. 10 to compute feasible control inputs to ensure safety at each time-step k. We initialize $I_a \leftarrow \emptyset$ and define $\Omega_{i\in I\setminus I_a} := \{u_o : (u_o - u_i)^T(u_o - u_i) \le \xi\}$. At each time-step k we maintain n_l state estimations for sensors in I_l and compute control input $u_i := \pi(\hat{x}_i)$ with a nominal controller. We compute u_o by solving (5.17), where J is a cost function. If no such u_o exists, we perform Alg. 9 and fault tolerant state estimation to remove conflicting sensors.

Theorem 5.3. *Given a safe set $\mathcal{C}$ and $\bar{\zeta}_s$, if the following conditions hold: (i) Assumption 1 holds, and (ii) scan match results r and $\tilde{r}$ can be found at each time step k, and (iii)*



(a) (b) (c) (d)

Figure 5.2: Comparison between the estimated LiDAR observations (blue lines) and actual LiDAR observations (pink lines). Fig. 5.2a to 5.2b compares the estimated and actual LiDAR observations under attack Scenario I (INS1 compromised). The estimate based on INS1 deviates from the actual scan, causing the compromised sensor INS1 to become untrusted. Fig. 5.2c to 5.2d compares the estimated and actual LiDAR observations under attack Scenario II (INS1 and LiDAR compromised). Fig. 5.2a and Fig. 5.2c estimate the LiDAR scan using the compromised measurements from INS1. Fig. 5.2b and Fig. 5.2d estimate the LiDAR scan using the measurements from INS2. The proposed approach removes the spoofed obstacle and aligns with the non-compromised sensor INS2.

there exists a function $B(x)$ satisfying the conditions in Proposition 5.1, then we have $Pr(x_k \in \mathcal{C},\ \forall 0 \leq k \leq T) \geq 1 - \gamma - cT$ when the adversary is present.

Proof. Given condition (i), (ii), and $\bar{\zeta}_s$, by Theorem 5.2, $\|x - \hat{x}_i\| \leq \bar{\zeta}_i$ for each sensor $i \in I \backslash I_a$. In Alg. 10, u is computed by a nominal controller and $\hat{u}$ is computed by solving (5.17). By condition (iii) and Proposition 5.1, we have $Pr(x[k] \in C, 0 \leq k \leq T_d) \geq 1 - \gamma - cT_d$. □

5.2.5 2D-LiDAR FTC Evaluation

This section evaluates our proposed approach on a UAV delivery system in an urban environment. The UAV system is based on MATLAB UAV Package Delivery Example [149]. The UAV adopts stability, velocity and altitude control modules, rendering its position control dynamics to be:

$$\begin{bmatrix}[x]_1 \\ [x]_2\end{bmatrix}_{k+1} = \begin{bmatrix}1 & -4.29 \times 10^{-5} \\ -1.47 \times 10^{-5} & 1\end{bmatrix} \begin{bmatrix}[x]_1 \\ [x]_2\end{bmatrix}_k + \begin{bmatrix}0.0019 & -1.93 \times 10^{-5} \\ -2.91 \times 10^{-4} & 0.0028\end{bmatrix} \begin{bmatrix}[u]_1 \\ [u]_2\end{bmatrix}_k, \quad (5.18)$$

where $x[k] = [[x]_1, [x]_2]^T$ is the UAV position, $[x]_1$ and $[x]_2$ represent the position of UAV on X-axis and Y-axis, respectively. The UAV has one LiDAR sensor and two inertial navigation system (INS) sensors, denoted as INS1 and INS2. The UAV maintains two EKFs associated with each INS sensor to estimate its position at each time k, denoted as $\hat{x}_1[k]$ and $\hat{x}_2[k]$, respectively. The system operates in the presence of an adversary who can compromise one of the INS sensors and spoof the LiDAR sensor.

We compare our proposed approach with a baseline utilizing a PID controller with state estimations given by INS1. We first demonstrate how our proposed approach selects sensors via Alg. 8 and Alg. 9 to obtain an accurate state estimation. We consider two attack scenarios. In Scenario I, the adversary compromises INS1 to deviate the measurement by -20 meters along the X-axis. In Scenario II, the adversary spoofs both the LiDAR sensor and INS1. The adversary biases INS1 sensor by -20 meters on X-axis and generates a random obstacle in the LiDAR scan within range of $[10, 15]$ meters and angle of $[-70, -60]$ degrees.

We present the estimated and actual LiDAR observations under Scenario I in Fig. 5.2a-5.2b. In Fig. 5.2a, we note that the estimated LiDAR observations $\mathcal{O}(\hat{x}_1, \mathcal{M})$ generated using state estimation $\hat{x}_1$ from INS1 significantly deviates from the actual LiDAR observations (the scan in pink color). The estimated LiDAR observations $\mathcal{O}(\hat{x}_2, \mathcal{M})$ align with the actual one as shown in Fig. 5.2b, which satisfies the criteria given in Section 5.2.3. Therefore, we treat INS2 as a trusted sensor while ignoring the measurements from INS1 when computing control input to the UAV.

We next compare the estimated and actual LiDAR observations under Scenario II in Fig. 5.2c-5.2d. The adversary manipulates the LiDAR observations by injecting a set of false

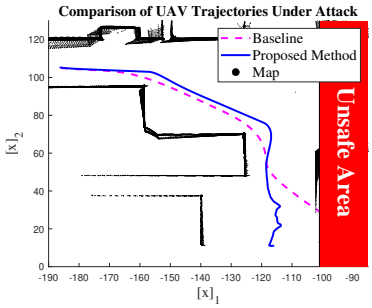


Figure 5.3: Comparison of trajectories of the UAV when controlled using our proposed approach and the baseline.

points around position $(5.5, -11.6)$. In Fig. 5.2c, we observe a significant drift between the estimated LiDAR observations $\mathcal{O}(\hat{x}_1, \mathcal{M})$ and actual LiDAR observations $\mathcal{O}(x, S)$. In Fig. 5.2b, the obstacle points contained in sector c generated by the LiDAR spoofing attack are eliminated by Alg. 9, and thus the estimated LiDAR observations $\mathcal{O}(\hat{x}_2, \mathcal{M}\backslash c)$ aligns with the LiDAR observations $\mathcal{O}(x, S\backslash c)$. In this case, our proposed fault tolerant estimation indicates that INS1 should be ignored and INS2 can be trusted.

We finally present the trajectories of the UAV with our proposed fault tolerant control (Alg. 10) and with the baseline. We present the trajectory of our proposed approach in Fig. 5.3 as the solid blue line, and the trajectory of the baseline as the dashed pink line. We observe that our proposed approach ensures the UAV to successfully avoid all obstacles and the unsafe area, whereas the baseline leads to safety violation due to lack of schemes to exclude faulty measurements.

5.3 Resilient Safe Control of 3D-LiDAR-based Systems

Autonomous vehicles rely on LiDAR sensors to detect obstacles such as pedestrians, other vehicles, and fixed infrastructures. LiDAR spoofing attacks have been demonstrated that either create erroneous obstacles or prevent detection of real obstacles, resulting in unsafe driving behaviors. In this section, we propose an approach to detect and mitigate LiDAR spoofing attacks by leveraging LiDAR scan data from other neighboring vehicles. This approach exploits the fact that spoofing attacks can typically only be mounted on one vehicle at a time, and introduce additional points into the victim's scan that can be readily detected by comparison from other, non-modified scans. We develop a Fault Detection, Identification, and Isolation procedure that identifies non-existing obstacle, physical removal, and adversarial object attacks, while also estimating the actual locations of obstacles. We propose a control algorithm that guarantees that these estimated object locations are avoided. We validate our framework using a CARLA simulation study, in which we verify that our FDII algorithm correctly detects each attack pattern.

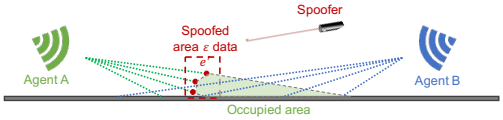
In what follows, we introduce the LiDAR observation and threat model that we consider in this section.

3D LiDAR Observation Model

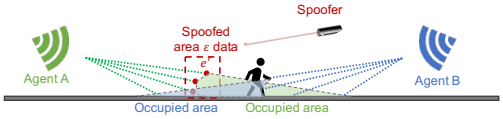
A LiDAR sensor fires and collects n_s laser beams and calculates the relative distance and angles to objects. For a laser beam indexed $i \in [1, n_s] \subseteq \mathbb{Z}^+$, we let $p_i := (s_i^r, s_i^a, s_i^\phi)$, where s_i^r denotes the range, s_i^a denotes the horizontal angle, and s_i^ϕ denotes vertical angle. A LiDAR sensor observes the environment by constructing a scan $S := \{p_i,\ 1 \leq i \leq n_s\}$. We denote the Cartesian translated LiDAR scan S measured at pose x as $\mathcal{O}(x, S) := \{(o_i^x, o_i^y, o_i^z), 1 \leq i \leq n_s\}$, where $o_i^x, o_i^y,$ and o_i^z denote the $x, y,$ and z coordinates of p_i. We assume that the vehicle has a default map, containing the locations of the infrastructure (for example, walls).

Threat Model

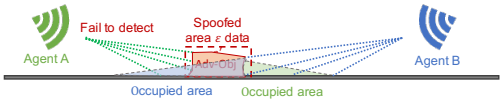
The purpose of the attacks is to disrupt the output of object detection algorithms of the LiDAR perception by either falsifying non-existing obstacles or hiding existing obstacles. Attacks on the positioning of the vehicle are out of scope. In this section, we consider three attacks with different purposes and implementation methods, as shown in Fig. 5.4.



(a) Relay attacks falsify non-existing obstacles.



(b) Physical removal attacks hide the object from the LiDAR detector.



(c) Adversarial objects can hide from the LiDAR detector.

Figure 5.4: Illustration of three attack types against LiDAR-based perception. Each attack type leaves a trace in the raw data that can be detected using our proposed approach.

Non-Existing Obstacle (NEO): The spoofer falsifies non-existing obstacles by introducing relay perturbations (Fig. 5.4a). In a relay attack, the adversary fires laser beams to inject artificial points e' into a LiDAR scan S. The resulting scan has points $S \cup e'$. Due to the physical limitation of the spoofing hardware, the injected point can only be within a very narrow spoofing angle. Hence, in this section, we assume that the relay adversary can only spoof one LiDAR sensor. As a result of the attack, the LiDAR detection algorithm incorrectly detects an obstacle between the spoofer and the LiDAR sensor.

Physical Removal Attack (PRA): As shown in Fig. 5.4b, the spoofer implements the physical removal attacks by sending a relay signal to the LiDAR receiver. The LiDAR detection algorithm believes that the true obstacle does not exist. In the scan S of the compromised LiDAR, there are artificial points e' between the true obstacle and the LiDAR. In order to realize the attack successfully, artificial points e' will reach the LiDAR receiver first and obscure the true obstacle. The true LiDAR signal reflected by the obstacle will be discarded by the LiDAR receiver. Due to the physical limitation of the spoofing hardware, only one LiDAR is compromised by the spoofer.

We further divide Attack PRA into three categories, based on whether the area containing the artificial points is fully observed by the uncompromised LiDAR sensor (PRA1), partially observed (PRA2), or not observed (PRA3).

Adversarial Object (AO): Adversarial objects are synthesized to be undetectable to the object detection algorithms of the LiDAR perception systems in a certain range of distance and angle (Fig. 5.4c). The adversarial objects introduce disturbance signal e' in the LiDAR scan S. More than one LiDAR sensor may be compromised by one adversarial object.

In this section, we assume that at most one attack occurs. The attack could be any of attacks NEO, PRA, or AO, and the autonomous vehicle (AV) does not know the attack type a priori.

5.3.1 3D LiDAR Fault Detection and Safe Control

In what follows, we propose a detection and control approach to ensure safety of the vehicle in an adversarial environment. The proposed approach is illustrated as Fig. 5.5. The victim agent leverages LiDAR observations from neighboring agents to detect and identify faults in two steps, namely, occupied area identification and the FDII, which are described as follows.

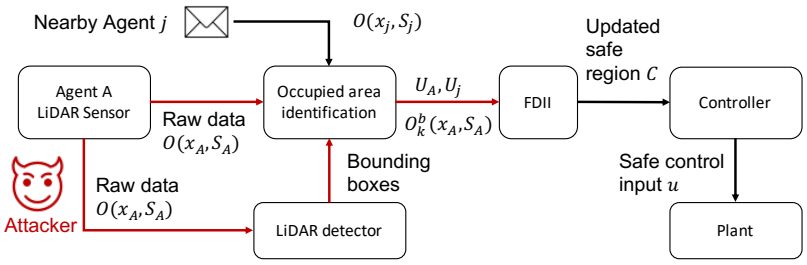


Figure 5.5: Schematic illustration of the proposed approach: to identify attacks marked in red, Agent A requests point cloud from nearby agents, i.e., Agent j. Then, FDII module takes $\mathcal{U}_A, \mathcal{U}_j, \mathcal{O}_k^b(x_A, S_A)$ and outputs detected attack type and updated safe region $\mathcal{C}$. Finally, controller output safe control input u.

Occupied Area Identification

In this subsection, we present our method to identify and extract the area that either contains obstacles or is not observable by the LiDAR, which we denote the occupied area. The detected occupied area will then be combined with the information from other vehicles to detect attacks and compute the unsafe region (Section 5.3.1).

We first prune the raw data of the observation $\mathcal{O}(x, S)$ by removing the points corresponding to the infrastructure (for example, walls) on a default map. For the rest of the paper, we use $\mathcal{O}(x, S)$ to denote the pruned observation. For Agent A, denote $L := \{j : \mathcal{I}_A \cap \mathcal{I}_j \neq \emptyset\}$ as the set of agents which have an overlapping scan-covered area with Agent A. We define the vertical projection operation $\mathcal{P}(\mathcal{O}(x, S)) \longrightarrow \mathcal{Y}$, which takes an observation $\mathcal{O}(x, S)$ as input and outputs the set $\mathcal{Y}$ of the projections of the points in $\mathcal{O}(x, S)$ onto the x-y coordinate plane. We define a non-obstacle scan-covered area $\mathcal{I}_j = \mathcal{P}(\bar{\mathcal{O}}(x, S))$ as the projection of the observation of Agent j, where $\bar{\mathcal{O}}(x, S)$ denotes the pruned observation within maximum LiDAR perception range.

We utilize the altitude coordinates of the points to distinguish the ground and the obstacle as described in [150][151]. We let ζ^z denote the estimation error of the altitude coordinate. The detection algorithm identifies $p_i = (o_i^x, o_i^y, o_i^z)$ as belonging to the ground if $o_i^z \leq \zeta^z$ and belonging to an obstacle if $o_i^z > \zeta^z$. After ground removal, we use the 3D bounding box provided by the LiDAR-based 3D object detection algorithms of the autonomous vehicles (AVs) [152] to cluster the points into a collection of obstacles, denoted $O_k^b(x, S) \subseteq O(x, S)$. Each obstacle has a corresponding bounding box denoted $\mathcal{B}_k \subseteq \mathbb{R}^3$. Here, the index k ranges

from 1 to n_o^j, where n_o^j denotes the number of obstacles detected by Agent j. Note that the set $O_1^b(x,S),\ldots,O_{n_o^j}^b(x,S)$ may contain fake obstacles introduced by an adversary. We regard the points that do not belong to any bounding box as the points introduced by Attack PRA or Attack AO. We further calculate the oblique projections $\mathcal{O}_k^p(x,S)$ of the points in $\mathcal{O}_k^b(x,S)$, which consists of the points where a straight line from the sensor to each point in $\mathcal{O}_k^b(x,S)$ intersects the ground. We have that $\mathcal{O}_k^p(x,S)\subseteq\mathbb{R}^3$.

We let $\mathcal{U}_{jk}$ denote the occupied area corresponding to Obstacle k observed by Agent j, which consists of the area that contains the obstacle as well as the area that is obscured by the obstacle. Formally, the occupied area $\mathcal{U}_{jk}$ is computed as the convex hull of $\mathcal{P}(\mathcal{O}_k^b(x,S))\cup\mathcal{O}_k^p(x,S)$. The convex hull can be calculated via open-source tools such as Scipy [153]. In order to compute the convex hull, we first obtain a collection of pairs of points $\{(x_1^l,y_1^l),(x_2^l,y_2^l):l=1,\ldots,n_{jk}^h\}$ from the object detection and oblique projection algorithms, where n_{jk}^h is the number of the boundaries of Obstacle k observed by Agent j. For each pair indexed l, we compute a half-plane constraint $h_l^{jk}(x)\leq 0$ with $h_l^{jk}(x)=a_l^{jk}\times x+b_l^{jk}$ where a_l^{jk} and b_l^{jk} are defined as $a_l^{jk}=\frac{y_2^l-y_1^l}{x_2^l-x_1^l}$ and $b_l^{jk}=\frac{x_2^l\times y_1^l-x_1^l\times y_2^l}{x_2^l-x_1^l}$. The convex hull is equal to the intersection of the half-plane constraints, i.e.,

$$\mathcal{U}_{jk}=\bigcap_{l=1}^{n_{jk}^h}\{x:h_l^{jk}(x)\leq 0\}.$$

The occupied area computed by the above procedure may not fully contain the obstacle due to the fact that the obstacle location must be interpolated from a finite number of samples. We enhance the robustness of these sampling errors by changing the boundaries of the occupied area to $h_l^{jk}(x)-||a_l^{jk}||\zeta_h\leq 0$, where $\zeta_h=\zeta_n+\zeta_r$, where ζ_n is the observation noise bound and ζ_r is a bound on the distance between neighboring sample points that can be obtained from the distance to the object and the angular resolution of the LiDAR. We assume that the LiDAR resolution is sufficiently large such that each point on the obstacle that is in the line-of-sight of the LiDAR is at most ζ_r distance away from at least one scan point.

In what follows, we will show that each obstacle visible to Agent j (including false or adversarial objects) is contained in an occupied region that is computed according to the procedure described above. Let $P^{ob}\subseteq\mathbb{R}^3$ denote the location of an obstacle. We divide the obstacle into two parts. The first part is the set of points in P^{ob} that have line-of-sight with

the LiDAR. We denote the first part as P_1. The second part, which is denoted as P_2, is the set of points in P^{ob} that do not have line-of-sight with the LiDAR.

Assumption 5.2. *For Agent $j \in L$ with occupied areas $\mathcal{U}_{jk}, k \in \{1, \ldots, n_o^j\}$ and Obstacle $k' \in \{1, \ldots, n_o^j\}$ with the set of points $P_{k'}^{ob} \subseteq \mathbb{R}^3$, we have $\mathcal{P}(P^{ob}) \bigcap (\cup_{k \in \{1,\ldots,n_o^j\} \setminus \{k'\}} \mathcal{U}_{jk}) = \emptyset$.*

Intuitively, Assumption 5.2 implies that for any obstacle visible to agent j, there is no overlap between the obstacle and the occupied area of any other obstacle detected by agent j.

Lemma 5.1. *Suppose that Assumption 5.2 holds and we are given the observation of an obstacle $\mathcal{O}_k^b(x, S)$ in a bounding box and the set of oblique projections $\mathcal{O}^p(x, S)$. If occupied area $\mathcal{U}$ is computed as the convex hull of $\mathcal{P}(\mathcal{O}_k^b(x, S)) \cup \mathcal{O}^p(x, S)$, then $\mathcal{P}(P^{ob}) \subseteq \mathcal{U}$.*

Proof. As described above, $P^{ob} = P_1 \cup P_2$. In what follows, we describe how $\mathcal{P}(P_1) \subseteq \mathcal{U}$ and $\mathcal{P}(P_2) \subseteq \mathcal{U}$, and hence $\mathcal{P}(P^{ob}) \subseteq \mathcal{U}$.

$\underline{P_1}$: Let $x \in P_1$. By assumption, since every point in P_1 is in the line of sight of the LiDAR, there exists a sample point x' such that $||x' - x|| \leq \zeta_h$. Hence, for all $l = 1, \ldots, n_{jk}^h$, we have

$$\begin{aligned} h_l^{jk}(x) &= a_l^{jk} x + b_l^{jk} \\ &= a_l^{jk}(x - x') + a_l^{jk} x' + b_l^{jk} \\ &\leq ||a_l^{jk}||\zeta_h + b_l^{jk} + a_l^{jk} x' \\ &\leq ||a_l^{jk}||\zeta_h \end{aligned}$$

where the first inequality follows from Cauchy-Schwartz and the second inequality follows from the fact that $h_l^{jk}(x') \leq 0$ for all scan points x'. Hence x lies in the occupied area.

$\underline{P_2}$: Suppose there is a point $p = (s^r, s^a, s^\phi) = (o^x, o^y, o^z) \in P_2$. There must exist a point $p_* = (s_*^r, s_*^a, s_*^\phi) = (o_*^x, o_*^y, o_*^z) \in P_1$ such that $s^a = s_*^a$, $s^\phi = s_*^\phi$ and $s^r > s_*^r$, which block the line-of-sight between the LiDAR and p. We denote the oblique projection of p and p_* as p_o. Since $s^a = s_*^a$ and $s^\phi = s_*^\phi$, the oblique projections of p and p_* are identical, which means that p, p_*, and p_o are collinear. Hence, the projection $\mathcal{P}(p)$, the projection $\mathcal{P}(p_*)$, and $\mathcal{P}(p_o)$ are colinear.

Finally, we show that $\mathcal{P}(p)$ belongs to the convex hull. Since the convex hull contains P_1 and its oblique projection, $\mathcal{P}(p_*)$ and p_0 belong to the convex hull. Since the convex hull is a

convex set, it contains the line segment $\overline{p_0\mathcal{P}(p_*)}$. Since the projection $\mathcal{P}(p)$, the projection $\mathcal{P}(p_*)$, and p_0 are collinear, $\mathcal{P}(p)$ belongs to the line segment $\overline{p_0\mathcal{P}(p_*)}$, which means that $\mathcal{P}(p)$ belongs to the convex hull.

Since both the projections of the points in P_1 and P_2 belong to the convex hull, and the occupied area $\mathcal{U}$ is computed as the convex hull of $\mathcal{P}(\mathcal{O}_k^b(x,S)) \cup \mathcal{O}^p(x,S)$, thus we have $\mathcal{P}(P^{ob}) \subseteq \mathcal{U}$. □

Fault Detection, Identification, and Isolation

The objective of the proposed fault detection, identification, and isolation (FDII) module is to detect the deviations between the observations of different agents, identify the spoofed agent, isolate the corrupted parts of the raw data, and restore the true unsafe region. The proposed FDII module is illustrated as Fig. 5.6. FDII first iterates over all detected obstacles to detect faults by leveraging LiDAR scan data from other neighboring vehicles. It then removes points contained in the bounding box and uses the residual point cloud for false data detection and identification.

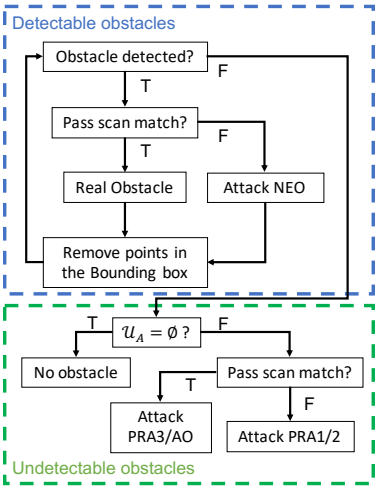


Figure 5.6: Decision tree of the FDII module: in the blue box, we iterate over detectable obstacles to detect faults. Then we remove points contained in bounding boxes and pass the remaining point cloud to the green box to identify undetectable obstacles.

We first describe how Agent A incorporates LiDAR information from other neighboring agents. At each time, Agent A collects the current observations of nearby agents $\mathcal{O}(x_j, S_j), \forall j \in L, j \neq A$, position x_j, and sensor information including observation noise bound and resolution bound.

The corresponding observations $\mathcal{O}(x_A, S_j)$, the observations of obstacles $\mathcal{O}_k^b(x_A, S_j)$, and the occupied areas $\mathcal{U}_{jk}$ are calculated by Agent A by repeating the steps described in Section 5.3.1. In $\mathcal{O}(x_A, S_j)$, Agent A marks the points in its bounding boxes as $\mathcal{O}_k^b(x_A, S_j) = \mathcal{O}(x_A, S_j) \cap \mathcal{B}_k$. For each $\mathcal{O}_k^b(x_A, S_j)$, Agent A calculates the corresponding $\mathcal{O}_k^p(x, S)$. Agent A then calculates $\mathcal{U}_{jk}$ by computing the convex hull of $\mathcal{P}(\mathcal{O}_k^b(x, S)) \cup \mathcal{O}_k^p(x, S)$.

We define the scan points in observation $\mathcal{O}(x_A, S_A)$ that is not detected by Agent A as $\mathcal{O}_A^u(x_A, S_A) := \mathcal{O}(x_A, S_A) \setminus \bigcup_k \mathcal{O}_k^b(x_A, S_A)$ and the undetected points in observation $\mathcal{O}(x_A, S_j)$ as $\mathcal{O}_j^u(x_A, S_A) := \mathcal{O}(x_A, S_j) \setminus \bigcup_k \mathcal{O}_k^b(x_A, S_j)$. We then compute their corresponding occupied area $\mathcal{U}_A^u$ and $\mathcal{U}_j^u$ by calculating their convex hulls.

We now analyze how each of the attacks defined in Section 5.3 affects the LiDAR perception and occupied area identification introduced in Section 5.3.1. In what follows, we use A to denote the index of the victim agent and analyze whether the obstacle observed by Agent A could also be observed and validated by other agents, i.e. B under each of the attacks defined in Section 5.3.

For Attack NEO, the artificial points e' introduce a fake obstacle with points $\mathcal{O}_k^b(x_A, S_A \cup e')$. Since there is no true obstacle, we have $\mathcal{U}_{Bk} = \emptyset$, and as a corollary, $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \nsubseteq \mathcal{U}_{Bk}$.

For Attack PRA1, the spoofed obstacle location does not overlap with the occupied area of agent B. Hence $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \nsubseteq \mathcal{U}_B^u$ and $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \cap \mathcal{U}_B^u = \emptyset$.

For Attack PRA2, there are also fake $\mathcal{O}_A^u(x_A, S_A \cup e')$ and non-empty $\mathcal{U}_B^u$. However, in this case

$$\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \nsubseteq \mathcal{U}_B^u, \text{ and } \mathcal{P}(\mathcal{O}_A^u(x_A, S_A)) \cap \mathcal{U}_B^u \neq \emptyset,$$

because Agent B could only observe the partial space of $\mathcal{O}_A^u(x_A, S_A \cup e')$.

For Attack PRA3 and AO, $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e'))$ is fully covered by non-empty $\mathcal{U}_B^u$, which means $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \subseteq \mathcal{U}_B^u$.

For the true obstacle, there is non-empty $\mathcal{O}_k^b(x_A, S_A \cup e')$ and non-empty $\mathcal{U}_{Bk}$ corresponding to the true obstacle. According to Lemma 5.1, we have $\mathcal{P}(P^{ob}) \subseteq \mathcal{U}_{Bk}$. Since $\mathcal{O}_k^b(x_A, S_A \cup e') \subseteq \mathcal{P}(P^{ob})$, we have $\mathcal{O}_k^b(x_A, S_A \cup e') \subseteq \mathcal{U}_{Bk}$.

The following lemma uses the preceding analysis to describe how the scan data from agent B can be used to detect and identify the attack type.

Lemma 5.2. *If* $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \setminus \mathcal{U}_{Bk} \neq \emptyset$, *then Agent A is being targeted by an attack of type NEO. If* $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \setminus \mathcal{U}_{Bk} = \emptyset$, *then Obstacle k is a true obstacle. If* $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \setminus \mathcal{U}_B^u \neq \emptyset$, *then Agent A is being targeted by PRA1 or PRA2. If* $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \setminus \mathcal{U}_B^u = \emptyset$, *then Agent A is under Attack PRA3 or AO.*

Proof. We first consider the detectable obstacles $\mathcal{O}_k^b(x_A, S_A \cup e')$. If $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \setminus \mathcal{U}_{Bk} \neq \emptyset$, which is equivalent to $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \neq (\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \cap \mathcal{U}_{Bk})$, then we have $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \nsubseteq \mathcal{U}_{Bk}$. According to the previous analysis of the impact of the attacks, Attack NEO occurs. If $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \setminus \mathcal{U}_{Bk} = \emptyset$ and $e' = \emptyset$, which is equivalent to $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) = (\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \cap \mathcal{U}_{Bk})$, then we have $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A \cup e')) \subseteq \mathcal{U}_{Bk}$. According to the previous analysis of the impact of the attacks, there is a true obstacle.

We next consider the undetectable obstacles $\mathcal{O}_A^u(x_A, S_A \cup e')$. If $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \setminus \mathcal{U}_B^u \neq \emptyset$, which is equivalent to $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \neq (\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \cap \mathcal{U}_B^u)$, then we have $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \nsubseteq \mathcal{U}_B^u$. According to the previous analysis of the impact of the attacks, Attack PRA1 or PRA2 occurs. If $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \setminus \mathcal{U}_B^u = \emptyset$, which is equivalent to $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) = (\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \cap \mathcal{U}_B^u)$, then we have $\mathcal{P}(\mathcal{O}_A^u(x_A, S_A \cup e')) \subseteq \mathcal{U}_B^u$. According to the previous analysis of the impact of the attacks, Attack PRA3 or AO occurs. □

We check whether $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \setminus \mathcal{U}_{Bk}$ is empty as follows. For each point $p \in \mathcal{P}(\mathcal{O}_k^b(x_A, S_A))$, we check to see whether p satisfies $h_l^{Bk}(p) - ||a_l^{jk}||\zeta_h \leq 0, \forall l \in \{1, \ldots, n_{Bk}^h\}$. If not, we put p into $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \setminus \mathcal{U}_{Bk}$. We name this operation a scan match.

We design the decision tree as shown in Fig. 5.6 to detect each attack. Agent A iterates the bounding boxes to check whether an object is detected. The points in the bounding box belong to either the non-existing obstacle from Attack NEO or the true obstacle. After detecting and identifying the attack or authenticating that it is the true obstacle, Agent A removes the points from the observation, then move on to the next bounding box. After traversing all bounding boxes iteratively, Agent A checks whether there is still observation of the obstacle. If not, there is no more attack or obstacle. If there is the observation of the obstacle, there is an attack and a scan match will be done to decide the classification of the attack.

After detecting the scan mismatch and identifying the spoofed agent, the proposed FDII algorithm isolates the corrupted parts of the raw data and updates the unsafe region by extracting the intersection of the unions of the occupied areas of the agents indexed in L.

The following theorem shows that the updated unsafe region $(\cup_{k=1,\dots,n_o^A}\mathcal{U}_{Ak})\bigcap(\cup_{m=1,\dots,n_o^j}\mathcal{U}_{jm})$ contains $\mathcal{P}(P_k^{ob}), \forall k \in \{1,\dots,n_o^A\}$. We note that Assumption 5.2 is not required in Theorem 5.4.

Theorem 5.4. *Suppose we are given the occupied areas $\mathcal{U}_{Ak}$ of Agent A and $\mathcal{U}_{jk}$ of Agent j, $k \in \{1,\dots,n_o^j\}$ in the area of $\mathcal{I}_A \cap \mathcal{I}_j$. If $\mathcal{P}(P_k^{ob}) \subseteq \mathcal{I}_A \cap \mathcal{I}_j$, then*

$$\mathcal{P}(P_k^{ob}) \subseteq (\cup_{k=1,\dots,n_o^A}\mathcal{U}_{Ak})\bigcap(\cup_{m=1,\dots,n_o^j}\mathcal{U}_{jm})$$

for any of the attack types NEO, PRA, or AO.

Proof. First, we show that even if there is another obstacle $P_{k'}^{ob}$ between P_k^{ob} and Agent j, $\mathcal{P}(P_k^{ob}) \subseteq \cup_k \mathcal{U}_{jk}$ holds.

For each obstacle $P_k^{ob}, \forall k \in \{1,\dots,n_o^j\}$, we consider two cases, namely (i) partially or fully blocked case, in which there is another obstacle $P_{k'}^{ob}$ between obstacle P_k^{ob} and Agent j, and $P_{k'}^{ob}$ partially or fully blocks P_k^{ob} from Agent j, and (ii) not blocked case, in which there is no $P_{k'}^{ob}$ that blocks P_k^{ob} from Agent j.

Case (i): In this case, there exists $k' \in \{1,\dots,n_o^j\}$ such that $\mathcal{P}(P_k^{ob}) \cap \mathcal{U}_{jk'} \neq \emptyset$. We divide P_k^{ob} into two parts: $\mathcal{P}(P_k^{ob}) \cap \mathcal{U}_{jk'}$ and $\mathcal{P}(P_k^{ob}) \setminus \mathcal{U}_{jk'}$. For $\mathcal{P}(P_k^{ob}) \cap \mathcal{U}_{jk'}$, we have

$$(\mathcal{P}(P_k^{ob}) \cap \mathcal{U}_{jk'}) \subseteq \mathcal{U}_{jk'} \subseteq \cup_k \mathcal{U}_{jk}.$$

For $\mathcal{P}(P_k^{ob}) \setminus \mathcal{U}_{jk'}$, since there is line-of-sight between $\mathcal{P}(P_k^{ob}) \setminus \mathcal{U}_{jk'}$ and Agent j, Assumption 5.2 holds. According to Lemma 5.1, we have

$$(\mathcal{P}(P_k^{ob}) \setminus \mathcal{U}_{jk'}) \subseteq \mathcal{U}_{jk} \subseteq \cup_k \mathcal{U}_{jk}.$$

Case (ii): In this case, $\mathcal{P}(P_k^{ob}) \cap \mathcal{U}_{jk'} = \emptyset, \forall k' \in \{1,\dots,n_o^j\}$. Since there is line-of-sight between $\mathcal{P}(P_k^{ob})$ and Agent j, Assumption 5.2 holds. According to Lemma 5.1, we have $\mathcal{P}(P_k^{ob}) \subseteq \mathcal{U}_{jk} \subseteq \cup_{m=1,\dots,n_o^j}\mathcal{U}_{jm}$.

We next show that even if one of the Attack NEO, PRA, or AO occurs, $\mathcal{P}(P_k^{ob}) \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}$.

For Attack NEO, $\mathcal{P}(P_k^{ob}) = \emptyset$, then $\mathcal{P}(P_k^{ob}) \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}$.

For Attack PRA, we first consider the case where e' is not blocked by any obstacle. According to the threat model in Section 5.3, we have e' obscuring P_k^{ob}. Hence according to Case (i), we have $\mathcal{P}(P_k^{ob}) \subseteq \mathcal{U}_{jk} \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}$. We next consider the case when e' is partially or fully blocked by another obstacle k'. According to Case (i), we have $\mathcal{P}(P_k^{ob}) \subseteq \mathcal{U}_{jk} \cup \mathcal{U}_{jk'} \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}$.

For Attack AO, obstacle P_k^{op} could be treated as a true obstacle with observation $\mathcal{O}(x_j, e')$. Hence, according to Case (i) and (ii), we have $\mathcal{P}(P_k^{ob}) \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}$.

Since $\mathcal{P}(P_k^{ob}) \subseteq \mathcal{U}_{jk} \subseteq \cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm}, j \in L$ holds for any of the attack types NEO, PRA, or AO, thus we have $\mathcal{P}(P_k^{ob}) \subseteq (\cup_{k=1,\ldots,n_o^A} \mathcal{U}_{Ak}) \bigcap (\cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm})$ for any of the attack types NEO, PRA, or AO. □

To calculate $(\cup_{k=1,\ldots,n_o^A} \mathcal{U}_{Ak}) \bigcap (\cup_{m=1,\ldots,n_o^j} \mathcal{U}_{jm})$, which is equivalent to

$$\bigcup_{k'=1,\ldots,n_o^A} (\mathcal{U}_{Ak'} \bigcap (\cup_{k=1,\ldots,n_o^j} \mathcal{U}_{jk})),$$

Agent A approximates the set $\mathcal{U}_{Ak'} \bigcap (\cup_{k=1,\ldots,n_o^j} \mathcal{U}_{jk})$ by computing the convex hull of all scan points whose projections are contained in $\mathcal{U}_{Ak'} \bigcap (\cup_{k=1,\ldots,n_o^j} \mathcal{U}_{jk})$. Then Agent A computes a set of half-plane constraints $h_l^{jk'}(x), l = \{1, \ldots, n_{jk'}^h\}$ based on the vertices, where $h_l^{jk'}(x) - ||a_l^{jk}|| \zeta_h < 0$ describes the corresponding half-plane and $n_{jk'}^h$ is the number of half-planes forming the convex hull $\mathcal{U}_{jk'}$. We define $\bar{h}_{jk'}(x) = \max_l h_l^{jk'}(x)$.

Safe Control

In this subsection, we present the safe vehicle controller based on the unsafe region provided by the FDII. The kinematic bicycle model [154] adopted in this paper is defined as

$$\begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{v}{l} \sin \psi \\ v \cos(\phi + \psi) \\ v \sin(\phi + \psi) \end{bmatrix} \tag{5.19}$$

where ϕ is the heading angle between the orientation and the x-axis, (x, y) is the position of the reference point (at the middle of the front axle, between the front wheels), v is the forward velocity at the reference point, l is the wheelbase, and ψ is the steering angle.

For implementation, we discretize Eq. 5.19 using forward differencing method with the sample time dt to obtain

$$\begin{aligned} \mathbf{x}[t+1] = & \begin{bmatrix} \phi[t+1] \\ x[t+1] \\ y[t+1] \end{bmatrix} = f(\mathbf{x}[t], \mathbf{u}[t]) \\ = & \begin{bmatrix} \phi[t] + dt \frac{v[t]}{l} \sin \psi[t] \\ x[t] + dtv[t] \cos(\phi[t] + \psi[t]) \\ y[t] + dtv[t] \sin(\phi[t] + \psi[t]) \end{bmatrix} \end{aligned} \tag{5.20}$$

where $\mathbf{u}[t] = (v[t], \psi[t])'$.

In order to avoid collision between the vehicle and the unsafe region, we utilize Model Predictive Control with Discrete-Time Control Barrier Function (MPC-CBF) [155]. The controller solves the following finite-time optimization problem with prediction horizon T at each time step

$$\min_{\mathbf{u}[t:t+T-1]} \quad (\mathbf{x}[t+T] - \mathbf{x}_r)^T F (\mathbf{x}[t+T] - \mathbf{x}_r) \tag{5.21a}$$

$$+ \sum_{t'=t}^{t+T-1} (\mathbf{x}[t'] - \mathbf{x}_r)^T Q (\mathbf{x}[t'] - \mathbf{x}_r) + \mathbf{u}[t']^T R \mathbf{u}[t']$$

$$\text{s.t. } \bar{h}_{jk'}(\mathbf{x}[t+1]) - \bar{h}_{jk'}(\mathbf{x}[t]) \geq -\gamma \bar{h}_{jk'}(\mathbf{x}[t]),$$

$$k' = 1, \ldots, n_o^A \tag{5.21b}$$

$$\mathbf{x}[t'+1] = f(\mathbf{x}[t'], \mathbf{u}[t']), t' = t, \cdots, t+T-1 \tag{5.21c}$$

$$\mathbf{x}[t'] \in X, t' = t, \cdots, t+T-1 \tag{5.21d}$$

$$\mathbf{u}[t'] \in U, t' = t, \cdots, t+T-1 \tag{5.21e}$$

where Eq. (5.21a) means that we would like the vehicle to converge to the midline of the lane (x_r) with minimal control effect, Eq. (5.21b) is the DT-CBF constraints ensuring that the vehicle avoids the unsafe regions provided by FDII, n_o^j is the number of the occupied areas of Agent j, Eq. (5.21c) is the system dynamics as shown in Eq. (5.20), Eq. (5.21d) is the admissible set of system state $\mathbf{x}$ (eg. stationary obstacles shown in the default map), and Eq. (5.21e) is the admissible set of control input $\mathbf{u}$ (eg. the upper bounds and lower bounds of the control inputs).

The optimal solution of Eq. (5.21) is a sequence of control inputs $\mathbf{u}^*[t], \ldots, \mathbf{u}^*[t+T-1]$. The first element $\mathbf{u}[t] = \mathbf{u}^*[t]$ will be executed. Since $\mathbf{u}[t] = \mathbf{u}^*[t]$ satisfies Eq. (5.21b), the vehicle will not collide with the obstacle at time step t. Then at time step $t+1$, Eq. (5.21) will be solved based on the new state $x[t+1]$. If FDII provides updated unsafe regions, they will be incorporated in Eq. (5.21b). This receding horizon control strategy guarantees collision avoidance between the vehicle and the obstacles.

5.3.2 3D-LiDAR FTC Evaluation

In this section, we evaluate our approach in CARLA [63] simulation environment. We first evaluate our proposed approach to FDII and obstacle detection under attacks. We then show that our safe controller ensures that the CARLA vehicle avoids obstacles using the proposed control strategy.

Augmented LiDAR FDII

We first introduce the simulation settings. We initialize two vehicles, denoted as Agents A and B, at locations $(-54.34, 137.05)$ and $(-34.34, 137.05)$, respectively. We evaluate our Augmented LiDAR FDII by simulating four scenarios: attack-free, attack NEO, PRA2 and PRA3.

We simulate the attack-free scenario as a reference group. As shown in Fig. 5.8a, a pedestrian spawns at $(-42.34, 137.05)$, 12 meters in front of Agent A.

In Attack NEO, the attacker injects false data to create a non-existing obstacle. As shown in Fig. 5.7a, the false data is injected into Agent A's LiDAR observation to falsify a cylinder 8 meters in front of agent A with radius 1 meter.

The PRA attacker spoofs LiDAR detection modules to hide obstacles in relay attacked area by injecting false data between victim agent and obstacles. We simulate this attack by replacing true measurement with Gaussian noise. In attack PRA2, we use the same basic setting as the aforementioned attack-free case. In addition, we let the attacker inject false data into Agent A's LiDAR observation to hide the pedestrian in the relay attacked area as shown in Fig. 5.8b. The injected false data, located 8 meters ahead of Agent A, is Gaussian noise distributed in a cylinder area with 1.3 meters radius.

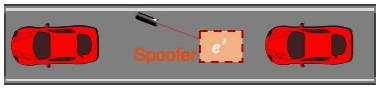
In the previous PRA2 case, the cylinder area can be partially seen by Agent B. We further validate our approach on a scenario where the cylinder area is blocked by pedestrian. In Attack PRA3 case, we set the cylinder area with 0.4 meter radius, shown in Fig. 5.8c.

Finally, we present the simulation results and analyze them by comparing with the reference attack-free scenario. We list the corresponding point cloud of joint perception of two agents and the intersection of the occupied area in the second and third row of Fig. 5.8, respectively.

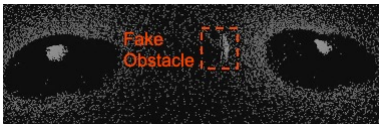
In the attack-free case, agents exchange point cloud data of the intersected area. Both agents can detect the obstacle and generate bounding boxes to contain those points. Agents identify occupied area and generate non-empty sets $\mathcal{U}_A$ and $\mathcal{U}_B$ with the contained points. The proposed FDII module takes these information and detects that there is no attack. By overlapping $\mathcal{U}_A$ and $\mathcal{U}_B$ according to agents' location, FDII further identifies the intersection shown in Fig. 5.8g as the candidate unsafe region.

In the attack NEO case, a fake obstacle is detected by Agent A annotated in Fig. 5.7b, which create an erroneous unsafe region. Due to the aforementioned limitation of relay attack, this fake obstacle can only be seen by Agent A. Therefore with the observation provided by Agent B, there is no occupied area identified with the points contained, i.e., $\mathcal{U}_B = \emptyset$. The proposed FDII module detects attack NEO and the obstacle is non-existing. Hence, the unnecessary unsafe region is removed, shown in Fig. 5.7c.

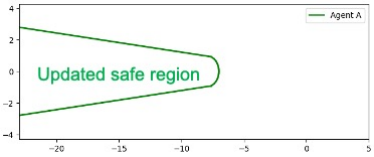
In attack PRA2 case, both attack signal and the pedestrian can be captured by Agent A and B respectively. With point cloud from Agent B, Agent A can observe both obstacles



(a) Attack NEO: An obstacle falsified by spoofer is set in front of Agent A.



(b) Fake obstacle can be detected only by Agent A with $\mathcal{U}_A \neq \emptyset$ and $\mathcal{U}_B = \emptyset$.



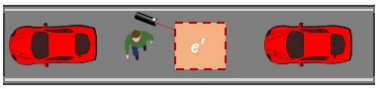
(c) The FDII module detects NEO attack. The intersection of the occupied area $\mathcal{U}_A \cap \mathcal{U}_B = \emptyset$, denoting there is no unsafe region.

Figure 5.7: FDII simulation settings and results of attack-NEO case

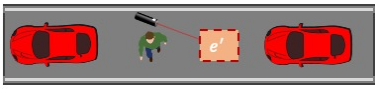
annotated in Fig. 5.8e and generate their corresponding occupied area with $\mathcal{U}_A \neq \emptyset$ and $\mathcal{U}_B \neq \emptyset$. Since some areas affected by injected false data can be observed by Agent B, we have the $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \nsubseteq \mathcal{U}_B$ and $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \cap \mathcal{U}_B \neq \emptyset$. The proposed FDII algorithm detects attack PRA2. By overlapping $\mathcal{U}_A$ and $\mathcal{U}_B$ according to agents' location, FDII further identifies the intersection shown in Fig. 5.8h as the candidate unsafe region.

In attack PRA3 case, the false data affected area is relatively small as shown in Fig. 5.8f, and hence the area is fully blocked by the pedestrian from Agent B's perspective. In this case, we have the $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \subseteq \mathcal{U}_B$ and $\mathcal{P}(\mathcal{O}_k^b(x_A, S_A)) \cap \mathcal{U}_B \neq \emptyset$. The proposed FDII algorithm detects attack PRA3. By overlapping $\mathcal{U}_A$ and $\mathcal{U}_B$ according to agents' location, FDII further identifies the intersection shown in Fig. 5.8i as the candidate unsafe region.

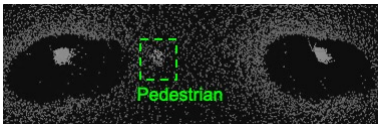
(a) Attack-free: a pedestrian is set between two agents without attack.



(b) Attack PRA2: The attack signal is set between Agent A and the pedestrian.

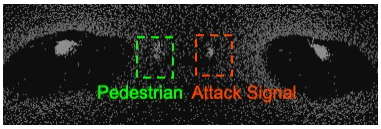


(c) Attack PRA3: The attack signal affects a relatively smaller area.

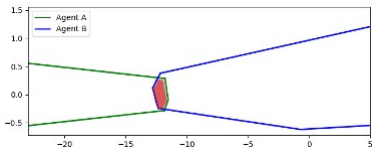


(d) Annotated pedestrian can be detected by both agents with $\mathcal{U}_A \neq \emptyset$ and $\mathcal{U}_B \neq \emptyset$.

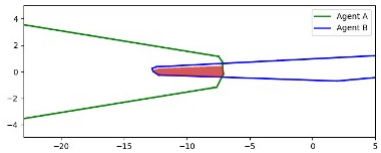
(e) Annotated attack signal is captured by Agent A with $\mathcal{U}_A \neq \emptyset$ and $\mathcal{U}_B \neq \emptyset$.



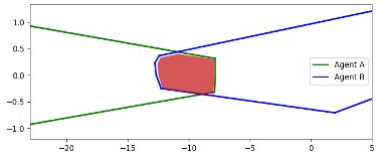
(f) Annotated attack signal is captured by Agent A with $\mathcal{U}_A \neq \emptyset$ and $\mathcal{U}_B \neq \emptyset$.



(g) The FDII module detects no attack. The intersection of the occupied area $\mathcal{U}_A \cap \mathcal{U}_B$ identified as an unsafe region.



(h) The FDII module detects PRA2 attack. The intersection of the occupied area $\mathcal{U}_A \cap \mathcal{U}_B$ identified as an unsafe region.



(i) The FDII module detects PRA3 attack. The intersection of the occupied area $\mathcal{U}_A \cap \mathcal{U}_B$ identified as an unsafe region.

Figure 5.8: Augmented LiDAR FDII simulation settings and results: We demonstrate settings in the first row, the corresponding point cloud of joint perception of two agents and the candidate unsafe region in the second and third row, respectively. We list attack-free, PRA2 and PRA3 in the three columns from left to right, respectively.

Safe Control

In this case study, we show that our MPC controller ensures the vehicle to be safe. We consider CARLA vehicle Model-3 as our control object and use (5.20) as the simplified vehicle model with $l = 4$ and $dt = 0.03$. We further take standard feedback linearization approach to acquire linear model as

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0.03 & 0 \\ 0 & 1 & 0 & 0.03 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_k \tag{5.22}$$

$$+ \begin{bmatrix} 0.0045 & 0 \\ 0 & 0.0045 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta v_x \\ \Delta v_y \end{bmatrix}_k, \tag{5.23}$$

in which v_x, v_y are velocity component of x-axis and y-axis, respectively, control input $u = [\Delta v_x, \Delta v_y]^T$ representing the corresponding changes.

We define an MPC controller according to (5.21) with F, Q, and R set to be identical matrices. We realize our controller with an open-source Python library do-mpc [156], which calls CasADi [157] and IPOPT [158] for nonlinear programming.

We next present the setting of the case. The vehicle is asked to perform reach-and-avoid task starting from location $(-14.34, 137.05)$ to $(-5.00, 135.25)$ without entering the unsafe region. We set the initial state to be $[-14.34, 137.05, 0, 0]^T$ and initial guess to be $[1, 0]^T$. Given the unsafe region detected by FDII module, controller restores the constraints on position $\boldsymbol{s} = [x, y]^T$ as

$$\begin{aligned} h(\boldsymbol{s})_1 &= [-0.35, 0.94]\boldsymbol{s} - 132.74 \\ h(\boldsymbol{s})_2 &= [-0.17, -0.99]\boldsymbol{s} + 132.5 \\ &\dots \\ h(\boldsymbol{s})_{14} &= [0.12, -0.99]\boldsymbol{s} - 134.62. \end{aligned}$$

Finally, we present the trajectory of the CARLA vehicle controlled by MPC in an urban street. As shown in Fig. 5.9, the vehicle drives from the location $(-14.34, 137.05)$ to $(-5.00, 135.25)$ without entering the unsafe region.

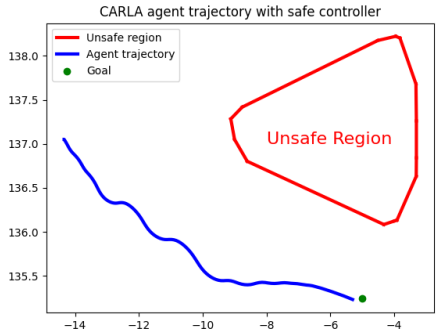


Figure 5.9: MPC drove CARLA vehicle from start $(-14.34, 137.05)$ to goal $(-5.00, 135.25)$. The agent managed to avoid detected unsafe region while tracking the given reference point.

5.4 Conclusion

In this chapter, we studied the problem of safety-critical control for a LiDAR-based system in the presence of sensor faults and attacks. We considered the class of systems equipped with a set of sensors for state and environment observations. We proposed a fault tolerant safe control framework for such systems to estimate their states and synthesize a control signal with safety guarantee. To obtain an accurate state estimate, we maintain a set of EKFs computed from different subsets of sensor measurements. For each estimate, we construct a simulated LiDAR scan based on the state estimates and an *a priori* known map, and exclude the state estimates that conflict with LiDAR measurements. When the LiDAR scan deviates from all of the state estimates, we remove the sector of the scan with the largest deviation. We proposed a control policy that selects a control input based on the fault tolerant estimate, and proved safety with a bounded probability using a control barrier certificate. We validated our proposed method with simulation studies on a UAV delivery system in an urban environment.

Later this chapter presented an approach for leveraging sensor data from neighboring vehicles to detect LiDAR spoofing attacks on autonomous vehicles. In our approach, vehicles exchange LiDAR scan data and identify spoofing attacks by checking for disparities between the detected

obstacles under each scan. We further develop a decision tree to differentiate between non-existing obstacle, physical removal, and adversarial object attacks. We then construct an estimate of the unsafe region based on the joint scan data, and propose a control policy that avoids the unsafe region. We validated our framework using the CARLA simulation platform and showed that it can detect and identify LiDAR attacks as well as guarantee safe driving.

Algorithm 8 LiDAR Scan Reconstruction

Input: State estimate $\hat{x}_i$, point-cloud map $\mathcal{M}$
Parameters: Resolution of the LiDAR scan c_r, maximum LiDAR range r_{max}.
Output: Estimated LiDAR Observation $\mathcal{O}(\hat{x}_i, \mathcal{M})$
Init: Set $\hat{x}_i$ as the center of scan $S_{\mathcal{M}}$, set $l_k^r \leftarrow r_{max}$. Separate the scan equally into $\frac{2\pi}{c_r}$ sectors S_k with corresponding angle l_k^a.
Translate points $m_j \in \mathcal{M}$ into polar coordinate with the origin $\hat{x}_i$, and represent it with a tuple (m_j^r, m_i^a).
for $m_j \in \mathcal{M}$ and $k \in [0, c_r]$ **do**
    if $m_j \in S_k$ and $m_j^r \leq l_k^r$ **then**
        $l_k^r \leftarrow p_i^r$
    end if
end for
for k *s.t.* $l_k^r = r_{max}$ **do**
    $l_k^r \leftarrow NaN$
end for
Reconstruct $S_{\mathcal{M}} = \{(l_k^r, l_k^a)\}$
Return $\mathcal{O}(\hat{x}_i, \mathcal{M}) = \mathcal{O}(\hat{x}_i, S_{\mathcal{M}})$

Algorithm 9 FT-LiDAR Estimation

Input: State estimation $\hat{x}$, number of sector n_j, Map $\mathcal{M}$ and LiDAR scan S
Output: r_j, c_j
Init: Equally separate scan S into n_j sectors $c_j \in S$
for $c_j \in S$ **do**
    Scan Reconstruction $\mathcal{O}_j(\hat{x}, \mathcal{M} \backslash c_j)$
    Scan Reconstruction $\mathcal{O}_j(x, S \backslash c_j)$
    Compute n_s^j the number of points in $S \backslash c_j$.
    Compute $\tilde{r}_j = \mathcal{O}_j(\hat{x}, \mathcal{M} \backslash c_j) \ominus \mathcal{O}_j(x, S \backslash c_j)$
    Compute $\zeta_s^j = n_s^j - \mathcal{L}_s(r_j)$
    if $\zeta_s^j \leq \bar{\zeta}_s$ **then return** $\tilde{r}_j, c_j$
    end if
end for

Algorithm 10 Fault Tolerant Control

Init: $I_a \leftarrow \emptyset$ and $\Omega_{i\in I\setminus I_a} := \{u_o : (u_o - u_i)^T(u_o - u_i) \leq \xi\}$
Maintain n_l EKFs for each sensor to estimate state $\hat{x}_i,\ i \in I_l = \{1, 2, \ldots, n_l\}$.
Compute control input $u_i := \pi(\hat{x}_i)$.
if control input $u \in \bigcap_{i\in I\setminus I_a} \Omega_i$ **then**
    set $\hat{u} = 0$ and $u_o = u + \hat{u}$
else ▷ STEP 1
    Compute control input $\hat{u}$ such that $u_o := u + \hat{u}$ is the solution to the following problem.

$$\min_{u_o} J(\hat{x}_i, u_o)\ s.t.\ u_o \in \cap_{i\in I\setminus I_a}\Omega_i \tag{5.17}$$

    if no such u_o can be found **then** ▷ STEP 2
        Perform FT-LiDAR Estimation (Alg. 9).
        Exclude false sensors into I_a by criteria I and II.
        Compute $\hat{u}$ by solving (5.17).
        if no such u_o can be found **then** ▷ STEP 3
            for $u \notin \bigcap_{i\in I\setminus I_a} \Omega_i$ **do**
                Compute residue values $y_i - o(\hat{x}_i)$
                Include i into I_a with the largest residue.
            end for
        end if
    end if
end if

Chapter 6

Conclusion

The rapid proliferation of autonomous systems into diverse and safety-critical domains, ranging from transportation and medicine to energy and robotics, underscores a paramount imperative: the formal assurance of their safe and reliable operation. As these systems become increasingly complex, integrating sophisticated sensors, actuators, and computational algorithms, any safety violation can precipitate catastrophic consequences, including substantial economic losses, severe injuries, or even the tragic loss of human lives. This thesis has confronted the central research challenge pivotal to realizing trustworthy autonomy: the end-to-end verification of complex autonomous systems.

Autonomous systems are fundamentally cyber-physical systems (CPS), characterized by a deep symbiosis between computational algorithms and physical components. The verification challenge, therefore, is not confined to software or hardware in isolation but spans the entire integrated system. For learning-enabled systems, this challenge is particularly formidable due to the inherent "black-box" nature of contemporary learning-enabled components. For real-world applications, autonomous systems encounter faults and cyber attacks, which lead to safety violations. This thesis approaches verifiably safe autonomy from two complementary directions: (i) safe control of learning-enabled systems providing formal guarantees, and (ii) resilient safe control that maintains formal safety guarantees under extreme scenarios such as sensor faults and cyber-physical attacks.

6.1 Advancing Verifiable Safety in Learning-Enabled Systems (Research Thrust 1)

A primary thrust of this dissertation has been to advance the formal verification of autonomous systems that integrate learning-enabled components, particularly those utilizing neural networks for control and decision-making. This endeavor has addressed systems with both deterministic and stochastic dynamics, consistently aiming for rigorous, provable safety guarantees.

For deterministic systems, as detailed in Chapter 2, a significant contribution lies in the development of exact safety conditions for NCBFs employing Rectified Linear Unit (ReLU) activation functions. Traditional verification methods often falter due to the non-differentiable nature of ReLU functions at certain points. By leveraging a generalization of Nagumo's theorem for proving invariance of sets with non-smooth boundaries, this work provides a sound theoretical basis for verifying such NCBFs. This theoretical advancement is complemented by a novel verification algorithm that exploits the piecewise-linear structure of ReLU networks, decomposing the NCBF into a collection of hyperplanes (differentiable segments) and hinges (non-differentiable intersections). To manage computational complexity, this verification strategically focuses on the boundary of the safe region. Furthermore, a VNN-based (Verification of Neural Networks) searching algorithm, utilizing Interval Bound Propagation (IBP) and linear relaxation, was introduced to efficiently identify these critical boundary hyperplanes and hinges.

Recognizing the computational demands of verifying each hyperplanes, the Synthesis with Efficient Exact Verification (SEEV) framework was proposed. SEEV integrates two key innovations: a training procedure incorporating a novel regularizer that penalizes the dissimilarity of activation patterns along the NCBF boundary, thereby reducing the number of hyperplanes to be verified; and an efficient verification algorithm employing a neural breadth-first search for enumerating boundary segments. This co-design of synthesis and verification, where safety counterexamples identified by the verifier are incorporated into the training dataset, has demonstrated significant improvements in verification efficiency and reliability. Experimental results shows substantial runtime reductions compared to state-of-the-art SMT-based methods like dReal and Z3, particularly for complex network architectures and higher-dimensional systems.

For stochastic systems, Chapter 3 extended the pursuit of verifiable safety by introducing Stochastic Neural Control Barrier Functions (SNCBFs). For smooth SNCBFs with twice-differentiable activation functions, verification was formulated as nonlinear programs solvable by SMT solvers. A step forward was the introduction of ReLU SNCBFs, for which sufficient safety conditions were derived using Tanaka's formula, again navigating the challenges of non-smoothness. Practical algorithms leveraging the piecewise linearity of ReLU networks were developed for efficient verification.

By directly addressing and improving the efficiency and scalability of verification techniques, this work makes tangible progress towards making end-to-end verification feasible for the increasingly complex learning-enabled systems envisioned for future autonomous applications. Futrue work could focus on the following directions:

- **Enhanced Scalability for Verification:** Developing novel algorithms, including advanced search techniques (e.g., refined branch-and-bound methods), to drastically improve the scalability of NCBF and SNCBF verification.
- **Verification for Diverse Neural Architectures:** Extending exact verification of NCBFs and SNCBFs to other activation functions (e.g., softplus, GELU) and network architectures (e.g., physical informed neural network) that are increasingly used in perception and control pipelines.

6.2 Enhancing Resilient Safe Control in Adversarial Environments (Research Thrust 2)

Chapter 4 addressed low-dimensional sensor faults and attacks, proposing High-Order Stochastic Control Barrier Functions (HOSCBFs) and Fault-Tolerant SCBFs (FT-SCBFs) tailored for systems with high relative degrees. These formulations are designed to ensure finite-time safety even when sensor readings are compromised. The feasibility of these FT-SCBFs, particularly those with high relative degrees, was rigorously established through Sum-of-Squares (SOS)-based verification schemes. The composition of HOSCBFs with Control Lyapunov Functions (CLFs) provides joint guarantees on both safety and stability under sensor fault conditions, ensuring that the system not only avoids unsafe regions but also progresses

towards its operational objectives. Recognizing the expressive power of neural networks, this work further introduced Fault-Tolerant Neural Control Barrier Functions (FT-NCBFs) for robotic systems. This involved deriving the necessary and sufficient conditions for FT-NCBFs to guarantee safety and developing a data-driven methodology to learn such FT-NCBFs by minimizing a loss function constructed from these formal conditions.

Chapter 5 shifted focus to the unique challenges posed by LiDAR perception attacks, which can critically undermine the environmental awareness of autonomous systems. For 2D-LiDAR systems, a fault-tolerant state estimation algorithm was developed, capable of reconstructing simulated scans from map data and state estimates to detect and remove false sensor inputs, including spoofed LiDAR measurements. This was coupled with a fault-tolerant safe control design using control barrier certificates, where SOS programs were employed to compute these certificates and verify safety constraints despite estimation errors induced by noise or attacks. The framework was validated on a UAV delivery system, which successfully avoided obstacles under attack conditions when using the synthesized control law, unlike baseline methods. For 3D-LiDAR systems, prevalent in autonomous vehicles (AVs), a novel safe control system was proposed that leverages cooperative perception, using point cloud data from neighboring vehicles. This system incorporates a sophisticated Fault Detection, Identification, and Isolation (FDII) module to detect, classify, and mitigate various LiDAR spoofing attacks, subsequently updating the unsafe region for the vehicle's planner. The correctness of this FDII module was analyzed, and its ability to accurately identify attack types and reconstruct the true unsafe region was validated through extensive simulations in the CARLA environment. The integrated system demonstrated successful navigation to a target while avoiding obstacles under attack.

This research thrust significantly broadens the scope of end-to-end verification by explicitly accounting for the imperfections and adversarial pressures characteristic of real-world operation. It demonstrates that verifiable safety is not limited to idealized, benign environments but can be systematically extended to systems facing component failures and malicious interference. A key element enabling this extension is the critical role of robust state estimation and sophisticated fault/attack detection mechanisms, such as the FDII module. These components ensure that the control loop operates on reliable or appropriately corrected information, making the subsequent safety guarantees meaningful and effective. Future work could explore the online adaptation of these safety mechanisms, allowing systems to dynamically adjust

their safety protocols in response to evolving threats or faults, as well as bridge the gap between simulation and real-world deployment.

- **Adaptive and Online Verifiable Resilience:** Moving beyond pre-defined fault or attack patterns to develop learning-enabled systems that can verifiably adapt their safety mechanisms online in response to novel or evolving faults and attacks. This would involve integrating online learning, anomaly detection, and rapid re-verification or robustification techniques.
- **Bridging Simulation-to-Real Gap with Formal Methods:** Investigating how formal verification techniques developed in simulation can provide stronger guarantees when controllers are deployed on physical hardware, explicitly accounting for model uncertainties, sensor noise characteristics, and hardware limitations within the verification framework.

References

[1] D. P. Watson and D. H. Scheidt, "Autonomous systems," *Johns Hopkins APL technical digest*, vol. 26, no. 4, pp. 368–376, 2005.

[2] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[3] A. M. Wyglinski, X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian, "Security of autonomous systems employing embedded computing and sensors," *IEEE micro*, vol. 33, no. 1, pp. 80–86, 2013.

[4] J. C. Knight, "Safety critical systems: Challenges and directions," in *24th International Conference on Software Engineering*, 2002, pp. 547–550.

[5] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[6] *Department of Homeland Security Cyber-Physical Systems Page*, Department of Homeland Security, Jan 2022, [Online] Available: https://www.dhs.gov/science-and-technology/cpssec [Accessed: Jan 2022].

[7] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1904–1911, 2022.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[9] S. Bansal, M. Chen, S. Herbert, and C. 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.

[10] M. Tayal, A. Singh, S. Kolathaya, and S. Bansal, "A physics-informed machine learning framework for safe and optimal control of autonomous systems," *arXiv preprint arXiv:2502.11057*, 2025.

[11] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier–value functions for safety-critical control," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6814–6821.

[12] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[13] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066.

[14] K. Lesser and M. Oishi, "Finite state approximation for verification of partially observable stochastic hybrid systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 159–168. [Online]. Available: https://doi.org/10.1145/2728606.2728632

[15] A. D. Ames, J. W. Grizzle, and P. 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.

[16] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, 2023.

[17] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3097–3110, 2020.

[18] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6105–6112.

[19] N. Jahanshahi, P. Jagtap, and M. Zamani, "Synthesis of partially observed jump-diffusion systems via control barrier functions," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 253–258, 2020.

[20] W. Xiao and C. 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.

[21] H. Dai and F. Permenter, "Convex synthesis and verification of control-Lyapunov and barrier functions with input constraints," *arXiv preprint arXiv:2210.00629*, 2022.

[22] S. Liu, C. Liu, and J. Dolan, "Safe control under input limits with neural control barrier functions," in *Conference on Robot Learning*. PMLR, 2023, pp. 1970–1980.

[23] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural Lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[24] D. R. Agrawal and D. Panagou, "Safe control synthesis via input constrained control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6113–6118.

[25] A. Clark, "A semi-algebraic framework for verification and synthesis of control barrier functions," *IEEE Transactions on Automatic Control*, 2024.

[26] S. Kang, Y. Chen, H. Yang, and M. Pavone, "Verification and synthesis of robust control barrier functions: Multilevel polynomial optimization and semidefinite relaxation," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 8215–8222.

[27] H. Dai and F. 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.

[28] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1364–1384, 2020.

[29] 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," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[30] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[31] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[32] T. Westenbroek, A. Agrawal, F. Castaneda, S. S. Sastry, and K. Sreenath, "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 19–24, 2021.

[33] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," *arXiv preprint arXiv:1901.10031*, 2019.

[34] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. 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. 11 532–11 539.

[35] M. Tayal, H. Zhang, P. Jagtap, A. Clark, and S. Kolathaya, "Learning a formally verified control barrier function in stochastic environment," *arXiv preprint arXiv:2403.19332*, 2024.

[36] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

[37] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, 2023.

[38] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock, "Learning safe neural network controllers with barrier certificates," *Formal Aspects of Computing*, vol. 33, pp. 437–455, 2021.

[39] H. Zhang, J. Wu, Y. Vorobeychik, and A. Clark, "Exact verification of ReLU neural control barrier functions," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[40] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. 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.

[41] A. Edwards, A. Peruffo, and A. 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.

[42] Y. Wang, C. Huang, Z. Wang, Z. Wang, and Q. Zhu, "Design-while-verify: correct-by-construction control learning with verification in the loop," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 925–930.

[43] H. Zhang, Z. Qin, S. Gao, and A. Clark, "Seev: Synthesis with efficient exact verification for relu neural barrier functions," *Advances in Neural Information Processing Systems*, vol. 37, pp. 101 367–101 392, 2025.

[44] A. Clark, "Control barrier functions for stochastic systems," *Automatica*, vol. 130, p. 109688, 2021.

[45] A. A. Amin and K. M. Hasan, "A review of fault tolerant control systems: advancements and applications," *Measurement*, vol. 143, pp. 58–68, 2019.

[46] J. Jiang and X. Yu, "Fault-tolerant control systems: A comparative study between active and passive approaches," *Annual Reviews in control*, vol. 36, no. 1, pp. 60–72, 2012.

[47] Y. Wang and X. Xu, "Observer-based control barrier functions for safety critical systems," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 709–714.

[48] E. Daş and R. M. Murray, "Robust safe control synthesis with disturbance observer-based control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 5566–5573.

[49] Y. Cheng, P. Zhao, and N. Hovakimyan, "Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 104–115.

[50] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Preprints of the 1st workshop on Secure Control Systems*, 2010, pp. 1–6.

[51] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security*, vol. 14, no. 1, p. 13, 2011.

[52] I. Punčochář, J. Širokỳ, and M. Šimandl, "Constrained active fault detection and control," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 253–258, 2014.

[53] Y. Shoukry, P. Nuzzo, A. Puggelli, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and P. Tabuada, "Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4917–4932, 2017.

[54] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[55] L. Niu, Z. Li, and A. Clark, "LQG reference tracking with safety and reachability guarantees under false data injection attacks," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 2950–2957.

[56] A. Clark, Z. Li, and H. Zhang, "Control barrier functions for safe CPS under sensor faults and attacks," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 796–803.

[57] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on LiDAR-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.

[58] Y. Cao, S. H. Bhupathiraju, P. Naghavi, T. Sugawara, Z. M. Mao, and S. Rampazzi, "You can't see me: physical removal attacks on LiDAR-based autonomous vehicles driving frameworks," *arXiv eprint archive*, 2022.

[59] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against LiDAR for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 445–467.

[60] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against LiDAR-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[61] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun, "Physically realizable adversarial examples for LiDAR object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 716–13 725.

[62] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li, "Invisible for both camera and LiDAR: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 176–194.

[63] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[64] W. Zhao, T. He, T. Wei, S. Liu, and C. Liu, "Safety index synthesis via sum-of-squares programming," *arXiv preprint arXiv:2209.09134*, 2022.

[65] M. Schneeberger, F. Dörfler, and S. Mastellone, "Sos construction of compatible control lyapunov and barrier functions," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 10 428–10 434, 2023.

[66] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[67] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," *arXiv preprint arXiv:2101.05436*, 2021.

[68] Z. Qin, T.-W. Weng, and S. 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. 12 903–12 910.

[69] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, 2020, pp. 1–11.

[70] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[71] Q. Zhao, X. Chen, Z. Zhao, Y. Zhang, E. Tang, and X. Li, "Verifying neural network controlled systems using neural networks," in *25th ACM International Conference on Hybrid Systems: Computation and Control*, 2022, pp. 1–11.

[72] K. Scheibler, S. Kupferschmid, and B. Becker, "Recent improvements in the SMT solver iSAT." *MBMV*, vol. 13, pp. 231–241, 2013.

[73] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5777–5783, 2018.

[74] M. Sha, X. Chen, Y. Ji, Q. Zhao, Z. Yang, W. Lin, E. Tang, Q. Chen, and X. Li, "Synthesizing barrier certificates of neural network controlled continuous systems via approximations," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 631–636.

[75] C. Ferrari, M. N. Muller, N. Jovanovic, and M. Vechev, "Complete verification via multi-neuron relaxation guided branch-and-bound," *arXiv preprint arXiv:2205.00263*, 2022.

[76] P. Henriksen and A. Lomuscio, "Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis." in *IJCAI*, 2021, pp. 2549–2555.

[77] H. Zhang, S. Wang, K. Xu, L. Li, B. Li, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "General cutting planes for bound-propagation-based neural network verification," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1656–1670, 2022.

[78] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 2017, pp. 97–117.

[79] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić *et al.*, "The Marabou framework for verification and analysis of deep neural networks," in *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*. Springer, 2019, pp. 443–452.

[80] F. B. Mathiesen, S. C. Calvert, and L. Laurenti, "Safety certification for stochastic systems via neural barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 973–978, 2022.

[81] R. Mazouz, K. Muvvala, A. Ratheesh Babu, L. Laurenti, and M. Lahijanian, "Safety guarantees for neural network dynamic systems via stochastic barrier functions," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9672–9686, 2022.

[82] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4939–4948, 2018. [Online]. Available: https://arxiv.org/pdf/1811.00866.pdf

[83] K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh, "Automatic perturbation analysis for scalable certified robustness and beyond," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1129–1141, 2020.

[84] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9835–9846, 2019.

[85] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C.-J. Hsieh, "Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=nVZtXBI6LNn

[86] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[87] H. Zhang, S. Wang, K. Xu, Y. Wang, S. Jana, C.-J. Hsieh, and Z. 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. 26 591–26 604.

[88] A. Edwards, A. Peruffo, and A. Abate, "Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models," *arXiv preprint arXiv:2311.09793*, 2023.

[89] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems," *arXiv preprint arXiv:2310.15478*, 2023.

[90] W. Zhao, T. He, T. Wei, S. Liu, and C. Liu, "Safety index synthesis via sum-of-squares programming," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 732–737.

[91] X. Wang, L. Knoedler, F. B. Mathiesen, and J. Alonso-Mora, "Simultaneous synthesis and verification of neural control barrier functions through branch-and-bound verification-in-the-loop training," in *2024 European Control Conference (ECC)*. IEEE, 2024, pp. 571–578.

[92] A. Peruffo, D. Ahmed, and A. 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.

[93] M. Anand and M. Zamani, "Formally verified neural network control barrier certificates for unknown systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 2431–2436, 2023.

[94] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Springer, 2008, vol. 78.

[95] J. Matousek and B. Gärtner, *Understanding and Using Linear Programming*. Springer Science & Business Media, 2006.

[96] X. Zeng, W. Lin, Z. Yang, X. Chen, and L. 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.

[97] A. J. Barry, A. Majumdar, and R. 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.

[98] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[99] C. Jewison and R. S. Erwin, "A spacecraft benchmark problem for hybrid control and estimation," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3300–3305.

[100] S. Gao, S. Kong, and E. M. Clarke, "dreal: An SMT solver for nonlinear theories over the reals," in *Automated Deduction–CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings 24*. Springer, 2013, pp. 208–214.

[101] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*. Springer, 2008, pp. 337–340.

[102] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[103] D. Revuz and M. Yor, *Continuous martingales and Brownian motion*. Springer Science & Business Media, 2013, vol. 293.

[104] C. Wang, Y. Meng, S. L. Smith, and J. 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.

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

[106] O. So, A. Clark, and C. Fan, "Almost-sure safety guarantees of stochastic zero-control barrier functions do not hold," *arXiv preprint arXiv:2312.02430*, 2023.

[107] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021.

[108] L. Ma and K. Khorasani, "Constructive feedforward neural networks using hermite polynomial activation functions," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 821–833, 2005.

[109] I. Karatzas and S. Shreve, *Brownian motion and stochastic calculus*. Springer Science & Business Media, 1991, vol. 113.

[110] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016.

[111] H. Zhang, L. Niu, A. Clark, and R. Poovendran, "Fault tolerant neural control barrier functions for robotic systems under sensor faults and attacks," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9901–9907.

[112] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and Fault-Tolerant Control*. Springer, 2006, vol. 2.

[113] Z. Chen, Y. Cao, S. X. Ding, K. Zhang, T. Koenings, T. Peng, C. Yang, and W. Gui, "A distributed canonical correlation analysis-based fault detection method for plant-wide process monitoring," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2710–2720, 2019.

[114] L. Li, H. Luo, S. X. Ding, Y. Yang, and K. Peng, "Performance-based fault detection and fault-tolerant control for automatic control systems," *Automatica*, vol. 99, pp. 308–316, 2019.

[115] A. M. Bardawily, M. Abdel-Geliel, M. Tamazin, and A. Nasser, "Sensors fault estimation, isolation and detection using MIMO extended Kalman filter for industrial applications," in *2017 10th international conference on electrical and electronics engineering (ELECO)*. IEEE, 2017, pp. 944–948.

[116] J.-S. Wang and G.-H. Yang, "Data-driven compensation method for sensor drift faults in digital PID systems with unknown dynamics," *Journal of Process Control*, vol. 65, pp. 15–33, 2018.

[117] D. Jung and E. Frisk, "Residual selection for fault detection and isolation using convex optimization," *Automatica*, vol. 97, pp. 143–149, 2018.

[118] J. Wang, C. Yang, H. Shen, J. Cao, and L. Rutkowski, "Sliding-mode control for slow-sampling singularly perturbed systems subject to markov jump parameters," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 12, pp. 7579–7586, 2020.

[119] Y. Yang, F. Chen, J. Lang, X. Chen, and J. Wang, "Sliding mode control of persistent dwell-time switched systems with random data dropouts," *Applied Mathematics and Computation*, vol. 400, p. 126087, 2021.

[120] Y.-A. Liu, S. Tang, Y. Liu, Q. Kong, and J. Wang, "Extended dissipative sliding mode control for nonlinear networked control systems via event-triggered mechanism with random uncertain measurement," *Applied Mathematics and Computation*, vol. 396, p. 125901, 2021.

[121] H. Yang, Y. Jiang, and S. Yin, "Fault-tolerant control of time-delay Markov jump systems with Ito stochastic process and output disturbance based on sliding mode observer," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5299–5307, 2018.

[122] H. Li, H. Gao, P. Shi, and X. Zhao, "Fault-tolerant control of Markovian jump stochastic systems via the augmented sliding mode observer approach," *Automatica*, vol. 50, no. 7, pp. 1825–1834, 2014.

[123] H. Yang, C. Huang, B. Jiang, and M. M. Polycarpou, "Fault estimation and accommodation of interconnected systems: a separation principle," *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4103–4116, 2018.

[124] Q. Zhao and J. Jiang, "Reliable state feedback control system design against actuator failures," *Automatica*, vol. 34, no. 10, pp. 1267–1272, 1998.

[125] X. Yu and Y. Zhang, "Design of passive fault-tolerant flight controller against actuator failures," *Chinese Journal of Aeronautics*, vol. 28, no. 1, pp. 180–190, 2015.

[126] Z. Li, L. Niu, and A. Clark, "LQG reference tracking with safety and reachability guarantees under unknown false data injection attacks," *IEEE Transactions on Automatic Control*, pp. 1–1, 2022.

[127] H. Heuser, *Lehrbuch der Analysis*. Springer-Verlag, 2013.

[128] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, "Stochastic stability of the continuous-time extended Kalman filter," *IEE Proceedings-Control Theory and Applications*, vol. 147, no. 1, pp. 45–52, 2000.

[129] J. Yin, S. Khoo, Z. Man, and X. Yu, "Finite-time stability and instability of stochastic nonlinear systems," *Automatica*, vol. 47, no. 12, pp. 2671–2677, 2011.

[130] Z. Chen, L. Li, and X. Huang, "Building an autonomous lane keeping simulator using real-world data and end-to-end learning," *IEEE Intelligent Transportation Systems Magazine*, 2018.

[131] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 322–328.

[132] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[133] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.

[134] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, 2020.

[135] J. Liu and J.-M. Park, ""Seeing is not always believing": Detecting perception error attacks against autonomous vehicles," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2209–2223, 2021.

[136] S. Wei, L. Ge, W. Yu, G. Chen, K. Pham, E. Blasch, D. Shen, and C. Lu, "Simulation study of unmanned aerial vehicle communication networks addressing bandwidth disruptions," in *Sensors and Systems for Space Applications VII*, vol. 9085. International Society for Optics and Photonics, 2014, p. 90850O.

[137] M. Hosseinzadeh, I. Kolmanovsky, S. Baruah, and B. Sinopoli, "Reference Governor-based fault-tolerant constrained control," *Automatica*, vol. 136, p. 110089, 2022.

[138] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on LiDAR-based perception in autonomous driving,"

in *ACM SIGSAC conference on Computer and Communications Security*, 2019, pp. 2267–2281.

[139] A. Khazraei, H. Pfister, and M. Pajic, "Resiliency of perception-based controllers against attacks," https://cpsl.pratt.duke.edu/sites/cpsl.pratt.duke.edu/files/docs/khazraei_ld4c22.pdf, Tech. Rep.

[140] R. S. Hallyburton, Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic, "Security analysis of camera-LiDAR fusion against black-box attacks on autonomous vehicles," *arXiv preprint arXiv:2106.07098*, 2021.

[141] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling UAVs with sensor input spoofing attacks," in *10th USENIX workshop on offensive technologies (WOOT 16)*, 2016.

[142] R. Matsumura, T. Sugawara, and K. Sakiyama, "A secure LiDAR with AES-based side-channel fingerprinting," in *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 2018, pp. 479–482.

[143] D. Suo, J. Moore, M. Boesch, K. Post, and S. E. Sarma, "Location-based schemes for mitigating cyber threats on connected and automated vehicles: A survey and design framework," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[144] T. Yang and C. Lv, "A secure sensor fusion framework for connected and automated vehicles under sensor attacks," *IEEE Internet of Things Journal*, 2021.

[145] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, "Stochastic stability of the discrete-time extended Kalman filter," *IEEE Transactions on Automatic control*, vol. 44, no. 4, pp. 714–728, 1999.

[146] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2003, pp. 2743–2748.

[147] A. Cotorruelo, M. Hosseinzadeh, D. R. Ramirez, D. Limon, and E. Garone, "Reference dependent invariant sets: Sum of squares based computation and applications in constrained control," *Automatica*, vol. 129, p. 109614, 2021.

[148] M. Kojima, S. Kim, and H. Waki, "Sparsity in sums of squares of polynomials," *Mathematical Programming*, vol. 103, pp. 45–62, 2005.

[149] "Matlab UAV package delivery," https://www.mathworks.com/help/uav/ug/uav-package-delivery.html.

[150] X. Qian and C. Ye, "NCC-RANSAC: A fast plane extraction method for 3-D range data segmentation," *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2771–2783, 2014.

[151] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3D laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[152] G. Zamanakos, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis, "A comprehensive survey of LiDAR-based 3D object detection methods with deep learning for autonomous driving," *Computers & Graphics*, vol. 99, pp. 153–181, 2021.

[153] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[154] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[155] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3882–3889.

[156] S. Lucia, A. Tătulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, pp. 51–62, 2017.

[157] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[158] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.