Towards Safer Autonomous Systems: From Vulnerability Mitigation to Safety Assurance

---

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Engineering)

by

Mahmoud Elnaggar

May 2024

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Engineering)

Mahmoud Elnaggar

Mahmoud Elnaggar

This dissertation has been read and approved by the Examining Committee:

Madhur Behl

Madhur Behl, Adviser

Zongli Lin

Zongli Lin, Committee Chair

James Smith

James Smith

Jack Davidson

Jack Davidson

Mircea Stan

Mircea Stan

Accepted for the School of Engineering and Applied Science:

Jennifer L. West

Jennifer L. West, Dean, School of Engineering and Applied Science

May 2024

To everyone who's helped me succeed

Contents

List of Tables

List of Figures

Abstract

Cyber physical systems (CPS) have become an integral part of our daily lives, from self-driving cars and autonomous delivery drones to industrial control systems. However, the safety of these systems remains a significant challenge due to the presence of cyber and system level vulnerabilities, unreliable wireless connectivity, and data-driven controllers. This dissertation proposes novel approaches as well as leverages existing ones in order to address the safety problem under three aspects.

The first aspect we address is the safety of vulnerable autonomous systems. Cyber security researchers have invented a myriad of techniques to protect against cyber attacks. With few exceptions, these techniques add runtime overhead to the system, which not only increases the time to complete any given task but more importantly might also put the systems under unsafe conditions when deployed to dynamical autonomous systems. We propose an adaptive algorithm that relies on model predictive control (MPC) to keep the system safe without taking unnecessarily conservative actions. We also consider system level attacks, i.e, a drone hijacking scenario where an attacker spoofs one or more of the onboard sensors of a drone to hijack it to an unsafe region. We propose an inverse reinforcement learning (IRL) based approach that predicts the intention of the attacker, determines the compromised sensor(s) and mitigates the attack.

The second aspect focuses on the safety of wirelessly connected autonomous systems. Connectivity between autonomous systems has multiple advantages in terms of performance and efficiency. However, for dynamical systems especially those operating in outdoor and urban environments, changes in the environmental context produce non-stationary effects on the wireless channel used for communication which might compromise the system safety. Our approach relies on a Baysian deep learning (BDL) model to predict the quality of the dynamical wireless channel in real time as well as the uncertainty of the model predictions. These predictions are key information needed by the control algorithms in order to take safe control actions that guarantee safety of the whole wirelessly connected system.

The third aspect of the dissertation addresses the safety of AI-controlled autonomous systems. We

present a provably safe neural network (NN) filter that filters any unsafe actions produced by a data-driven Reinforcement Learning (RL) controller and guarantees that the system state will always remain inside a safe set. The approach comprises designing, verifying and synthesizing a control barrier function (CBF) based on a kinematics model of the system. The merit of the proposed approach is that it allows us to decouple the handling of CBF constraints from the control optimization task while having a computationally efficient yet provably safe control action filter.

The presented approaches are validated through case studies, simulations and experiments, demonstrating their effectiveness in ensuring safety of CPS under various scenarios. The research contributes to the field of CPS by providing comprehensive techniques to solve safety problems, which can be applied to various types of autonomous systems.

Keywords: Reinforcement Learning (RL); Inverse Reinforcement Learning (IRL); Control Barrier Function (CBF); cyber-physical systems security; autonomous driving safety; Model Predictive Control (MPC); neural networks verification; Bayesian Deep Learning (BDL).

Chapter 1

Introduction

Nowadays, modern vehicles are becoming autonomous primarily thanks to the recent breakthroughs in machine learning coupled with the large use of high performing embedded computers and sensors. This increased use of automation has however opened the door to multiple cyber-security attacks. In fact, any computing system that accepts inputs that can potentially be manipulated by a malicious adversary is potentially vulnerable to cyber attack. For example, autonomous vehicles can be subject to wireless attacks in which an adversary uses a wireless communication channel to send malicious inputs (e.g., control inputs, map updates, mission updates, etc.) to hijack the system and exploit vulnerabilities in the system [2].

Recent advancements in cybersecurity have introduced sophisticated techniques aimed at safeguarding systems and data from cyber threats. However, these techniques often come with a runtime overhead that can potentially compromise the performance and safety of cyber-physical systems (CPS). Machine learning algorithms, for instance, have been increasingly employed in cybersecurity for anomaly detection and threat identification [3]. While effective, the computational complexity of these algorithms can lead to significant runtime overhead, particularly in real-time CPS applications where timely responses are critical. Behavioral analysis is another cybersecurity technique that relies on monitoring system behavior to detect abnormal activities indicative of a security breach [4]. While proven effective, the continuous monitoring required for behavioral analysis can introduce substantial runtime overhead, impacting the responsiveness of CPS. Additionally, encryption and cryptographic techniques, vital for securing sensitive information, often incur computational overhead during encryption and decryption processes [5]. In CPS, where resource-constrained devices are prevalent, excessive computational overhead from encryption can hinder real-time operations and compromise system safety. The adoption of Zero Trust Architecture (ZTA) in CPS environments

introduces runtime overhead associated with continuous authentication and access validation mechanisms [6]. While ZTA enhances security by assuming no trust, the additional processing required for authentication can delay critical operations in CPS, potentially endangering system safety. Similarly, containerization and microservices, which offer scalability and agility benefits in software development, introduce security challenges and runtime overhead in CPS environments [7]. The dynamic nature of containerized environments requires constant monitoring and enforcement of security policies, adding computational burden and latency to CPS operations. Secure Multi-Party Computation (SMPC), although promising for facilitating secure collaborative computations, imposes computational overhead due to cryptographic operations [8]. In CPS, where real-time coordination among distributed components is crucial, excessive computational overhead from SMPC can disrupt system responsiveness and compromise safety-critical tasks. Hardware-based security mechanisms like Trusted Platform Modules (TPMs) and Hardware Security Modules (HSMs) provide secure execution environments but add overhead in key management and secure bootstrapping processes. In CPS, where timing constraints are stringent, the additional overhead from hardware-based security can lead to missed deadlines and system failures. When such techniques are deployed on autonomous dynamical systems that interact with their environments (e.g., drones, cars, robots), it is necessary to adapt the control performance to avoid unsafe conditions due to the delays introduced by the security mechanisms. For example, controllers are built to run at a given rate. During normal execution, inputs are applied for a certain duration. A delay may result in an erroneous application of the same input for longer times which can create unsafe conditions. For example, a vehicle may speed up or collide with other obstacles or vehicles.

Moreover, the majority of modern cyber-physical systems (CPSs) are not built with cybersecurity in mind. The tight coupling between information technology and the physical world have introduced security vulnerabilities in the form of physical and cyber-attacks (e.g., sensor, actuator, controller, communication and environment attacks).

Multiple incidents that have occurred recently indicate the ability of a malicious attacker to compromise modern cyber-physical systems using different techniques like the the well-known StuxNet cyber-attack on an Iranian nuclear reactor [9] and multiple cyber-attack demonstrations on modern automobiles [10].

For example, it is believed that GPS spoofing led to the capturing of a sentinel drone in Iran in 2011 [11]. More recently, researchers in [12] have also demonstrated a GPS spoofing attack on a vessel, making it deviate from its desired course.

The key insight of the first part of this research is that many cyber security techniques can be used to enhance autonomous vehicle (AV) security, if the vehicle is capable of dealing with the overhead implications.

To realize this possibility, we address the implications of applying security techniques on autonomous vehicle control systems by adapting low-level controllers to maintain the required system safety properties. Secondly, we focus on handling safety issues that come as a byproduct of system-level attacks, specifically sensor spoofing attacks in which a malicious program masquerades as another sensor by falsifying data while staying stealthy within the error noise of the sensor measurement and the uncertainties of the system. GPS spoofing in particular has been demonstrated in several works [11–13]. A malicious attacker can use a hand-held device to deceive a GPS receiver by broadcasting signals synchronized with the genuine signals observed by the target receiver. The power of the counterfeit signals is then gradually increased and drawn away from the genuine signals [13]. Such an attack has been demonstrated on different vehicles compromising their safety and/or hijacking them to undesired states. Particularly, we consider the case where an autonomous vehicle, equipped with multiple sensors, is tasked to perform a go-to-goal navigation. A malicious attacker performs a coordinated attack by spoofing one or more of the sensors on the vehicle. The goal of the attacker is to hijack the vehicle to an undesired goal (not known beforehand) while hiding within the sensor and actuator noise profile and disturbance model of the system. Our goal is to develop a technique to infer the intention of the attacker and recover the system before reaching the undesired state.

In other words, the first part of this thesis tries to answer the following research questions:

- How to design a controller that guarantees safety of an autonomous system equipped with cyber-security tools that incur runtime overheads while also maintaining the desired system performance?
- Considering the drone hijacking scenario described above, how can we i) predict the intention of the attacker ii) determine the set of compromised sensors and iii) recover the uncompromised state of the drone and drive it back to its desired objective?

The second part of this thesis focuses on safety aspects of wirelessly connected autonomous systems. Wireless connectivity between cooperating autonomous systems can greatly enhance their performance and capabilities. For example, vehicle platoons increase both highway throughput and fuel efficiency by traveling nearly bumper-to-bumper, using a wireless coupling to brake and accelerate simultaneously [14]. Vehicles or drones can move around blind corners at high speed by leveraging the sensing capabilities of the agents ahead of them [15]. Figure 1.1 illustrates an aspirational example of the above two notions–increasing capacity and performance while leveraging (uncertain and changing) communication.

Wireless coordination could enable mobile systems to reach high performance states that would not otherwise be safe (closer distances, higher speeds, etc). However, there is currently no methodology for providing provable safety guarantees for such states. The main challenge is to capture the interdependence

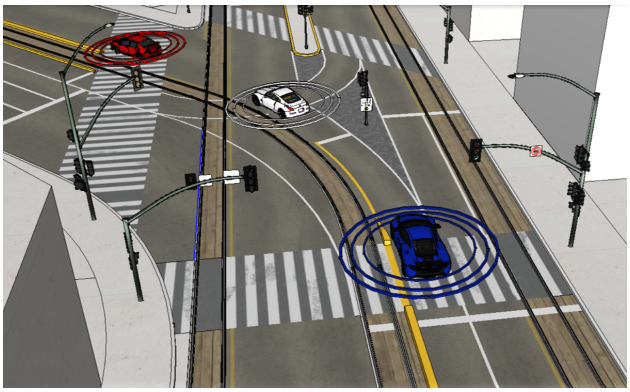


Figure 1.1: Motivating Example. Our approach develops techniques where turning at a faster speed can actually be safer by maintaining wireless communication with the agent ahead.

between mobility, wireless, and safety: an agent's motion has a profound effect on the wireless channel, which in turn affects the ability to maintain physical safety.

Wireless signals are transmitted through unguided media such as air or water and are therefore greatly affected by the surrounding environment. As mobile agents move through a physical environment, especially complex indoor environments but also outdoor and urban environments, changes in the environmental context produce *non-stationary* effects on the channel: future channel properties are not well predicted by those of the past. Despite this, formal methods currently model networked systems by assuming a stationary communication channel. This stationarity assumption first took root in the analysis of wired communication systems [16]. As these methods were generalized to wireless systems, the assumption has continued to hold true for slow control loops that tolerate long network latencies (e.g. minutes), including many aerospace [17] and vehicle applications [18]. This is because the long-term average properties of a wireless channel are likely to be fairly stable. However, as wireless becomes part of faster control loops with shorter time constants, such as wirelessly coordinated multi-agent mobility, this assumption no longer holds.

There have been many recent developments for the synthesis and formal analysis of control and communication schemes for networked, multi-agent mobile systems, but these techniques currently assume a stationary communication channel. Given fixed communication requirements (e.g. packet delivery rate), any stationary channel model translates to a bound on the maximum separation D between agents: there is no guarantee of communication beyond distance D, and there is no benefit to moving agents closer than D. Therefore, these methods can only currently reason about communication requirements as distance-based connectivity requirements. Prior work has also demonstrated the ability to produce connectivity guarantees by synthesizing control strategies from formal specifications [19]. However, many of these assumptions ignore the effect of mobility and dynamic environments on the wireless channel; therefore, the controller cannot

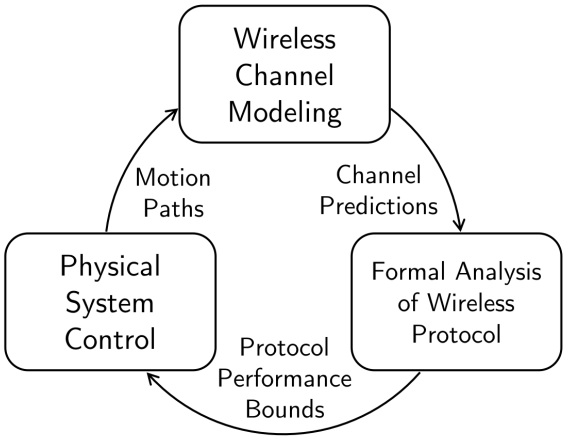


Figure 1.2: Overview of the proposed approach and connections between three problems.

choose a motion plan that simultaneously assures safety and improves performance, which is supposed to be one of the promises wireless communication.

Therefore, three broad requirements motivate the design of safety-critical wireless mobile systems:

1) Non-stationary channel: In a dynamic environment with dynamic agents, the system(s) must be able predict properties of a non-stationary channel with only partial models of the physical world. Furthermore, in safety-critical applications, the system must be able to assess the certainty of these predictions.

2) Worst-case performance bounds: Many approaches to wireless communication modeling and prediction attempt to provide average-case performance predictions. To achieve a vision of safety-critical wireless, it is crucial to efficiently compute wireless performance *bounds* at runtime using predictions of requirement 1.

3) Safety guarantees with performance improvements: Given the potential benefits of wireless connectivity, the system must determine what information needs to be communicated in order to simultaneously assure safety and significantly improve other mission objectives. It must then use this information in a scalable, computationally efficient way to generate optimal, safe trajectories in a multi-agent setting.

Our contributions achieve these requirements by trying to answer the following questions:

- How can we estimate characteristics of non-stationary wireless channels with only partial models of the physical world and predicted motion paths of mobile vehicles?
- How to incorporate these estimations into the formal analysis of wireless protocols to compute the worst-case (or best) wireless performance bounds at the runtime? rather than just average-case performance.

- How to use these bounded latency predictions to plan and execute trajectories for mobile vehicles that provide probabilistic guarantees on safety while still improving the overall performance of a multiple mobile agents?

The third part of this thesis focuses on safety assurance of NN controlled autonomous systems. Deep Neural Networks (DNNs) are increasingly popular in robotics control tasks, especially as a means of parameterizing the optimal policy in Deep Reinforcement Learning (DRL). However, the ability to obtain *provably* safe data-trained NN controllers has not kept pace with their increasing deployment in safety-critical applications such as autonomous vehicles. In this paper, we thus introduce a new approach to the design of safe data-trained feedback controllers[1] for such autonomous (four-wheeled) vehicles.

In particular, we propose ShieldNN, an algorithm to design Rectified Linear Unit (ReLU) networks with provable obstacle avoidance properties for the Kinematic Bicycle Model, which is a good approximation for four-wheeled vehicles [20]. Moreover, this safety is obtained by means of a unique architecture and deployment: a ShieldNN network composed in series with *any* memoryless feedback controller[1] ensures that the composition of the two controllers has provable obstacle avoidance properties. This structure distinguishes ShieldNN from most other work on safe data-trained controllers: instead of designing a single safe controller, ShieldNN uses the KBM dynamics to design a *controller-agnostic* NN that corrects – in real-time – unsafe control actions generated by *any* controller. That is ShieldNN designs a "safety-filter" NN which takes as input the instantaneous control action generated by a controller (along with the state of the system) and outputs a safe control action for the KBM dynamics; this safety-filter NN thus replaces *unsafe* controls generated by the original controller with safe controls, whereas *safe* controls generated by the original controller are passed through unaltered – i.e., unsafe controls are "filtered" out. A block diagram illustrating the use of a ShieldNN filter is illustrated in 1.3. The benefits of this approach are manifest, especially for data-trained controllers. On the one hand, existing controllers that have been designed *without* safety in mind can be made safe by merely incorporating the safety filter in the control loop. In this scenario, the safety filter can also be seen as a countervailing factor to controllers trained to mimic experts: the expert learning can be seen as a design for "performance", and the safety filter is added to correct unanticipated unsafe control behavior as needed. On the other hand, the controller-agnostic nature of our proposed filter means that ShieldNN itself may be incorporated into training. In this way, the safety filter can be seen to function as a kind of "safety expert" during training, and this can potentially improve sample efficiency by eliminating training runs that end in unsafe states. We can summarize the research questions this part of the thesis tries to answer as follows:

[1] RL-trained feed-forward neural networks, for example.

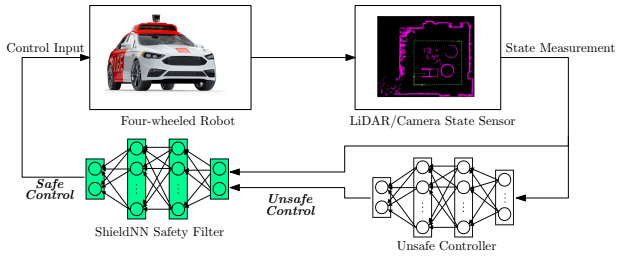


Figure 1.3: Block diagram of ShieldNN in the control loop for a four-wheeled autonomous vehicle.

- Can we design a candidate barrier function for a four wheeled ground vehicle?
- How to verify the existence of safe control given the candidate barrier function?
- How to design and synthesise a control filter based on the candidate barrier function?

Chapter 2

Related work

2.0.1 CPS Security

In recent years there has been a growing interest by the research community in the security of AVs. Kim et al. discuss the security vulnerabilities of AVs [2]. Specifically, they note that wireless communication channels provide one means for attack, including gaining full control of the AV by exploiting buffer overflow vulnerabilities [21]. Javaid et al. develop a cyber security threat model for AVs [22]. They discuss integrity attacks that include the use of malicious code or exploiting existing subroutines. In discussing the technical challenges of securing AVs, Wyglinski et al. [23] discuss remote access vulnerabilities which includes, for example, introducing malicious code into a system through an update.

To counter attacks against systems, security researchers have developed numerous techniques to prevent attacks. Abadi et al. discuss control-flow integrity, a powerful technique to prevent control hijacking attacks [24]. The use of various randomization techniques to introduce artificial diversity has become a standard and widely deployed cyber defense [25, 26]. The use of intrusion detection systems [27, 28] and anti-virus systems [29, 30] are also widely used to prevent various types of intrusions and attacks.

From a control perspective, recent research on cyber-physical systems (CPS) has tackled security problems on modern vehicles and autonomous robotic systems considering sensor, actuator, communication, controller, and physical attacks. Recent incidents that illustrate the susceptibility of CPS to attack include the StuxNet attack on an industrial SCADA system [31] and attacks on automobiles [32], vessels [33], and military drones [34] in which systems may be tampered with, yielding unexpected behavior.

Hence, there is a need to design and develop systems that can tolerate, prevent, and detect attacks and recover and reconfigure to guarantee safety, and in recent years we have witnessed an increased academic

effort to address such cybersecurity issues. Most of the literature and current research, however, focuses on how to detect and estimate malicious attacks considering partial or full observability of the internal state of a system [35–37] and does not deal with the problem of adapting the controller, which is the subject of this work.

Consequently, significant research effort has been applied to develop control-level techniques that exploit knowledge of system dynamics to address the problem of attack-resilient state estimation, as well as intrusion detection under different types of attacks on sensors, actuators and communication networks while considering robust controllers like PID regulators [36, 38–40]. Another example is the on-going DARPA High-Assurance Cyber Military Systems (HACMS) program [41], devoted to the design of secure control systems for autonomous vehicles. In HACMS, Bezzo and his colleagues developed techniques for resilient state estimation and resilient sensor fusion that can tolerate various kinds of sensor attacks.

The literature on CPS system level cybersecurity has been growing substantially in the last few years. Among the earliest work on this domain we find [38, 42–44] in which resilient state estimators (RSEs) were developed to leverage redundancy in sensor data and knowledge about the CPS dynamics. One of the most significant results obtained in [42, 44] shows that by means of such RSEs it is possible to correctly reconstruct the state of the system if the number of attacked sensors is bounded. Continuing from a control point of view, in [45] a convex optimization algorithm was proposed to minimize estimation error and control an electric vehicle in which communication was not reliable due to a malicious cyber-attack. In [46] a satisfiability modulo theory-based approach was presented to detect and estimate the state of an unmanned aerial vehicle (UAV) under sensor attack. Remaining on UAV applications, authors in [47] presented a software architecture that allows a UAV to operate with compromised system components by virtualizing sensor, actuators, and communication channels and switching to a safe behavior once an attack is detected.

Following a machine learning approach, in [48] the authors researched the problem of attack detection and reconfiguration on industrial control systems by proposing a data-driven machine learning approach to detect anomalies in sensor readings. In our previous work [49] we introduced a Redundant Observable Markov Decision Process (ROMDP) approach to deal with sensor attacks in a stochastic environment. ROMDP selects the optimal policy to use in order to minimize the probability of reaching undesired states in the environment. All of the aforementioned works are concerned with detecting an attack, estimating the state of the system, and recovering the system. Instead, in this work, we are concerned with the problem of inferring the intention of an attack, a problem that to the best of our knowledge has not been investigated in the current CPS-security literature. To this end, we leverage Inverse Reinforcement Learning (IRL) techniques to

predict the intended goal of an attacker that is spoofing one or more sensors to hijack a UAV from its desired route. Inverse Reinforcement Learning technique was first introduced in [50]. Authors in [51] introduced the Bayesian Inverse Reinforcement Learning (BIRL) algorithm for reward and policy learning. This work on BIRL was extended in [52] to deal with continuous state and action spaces leveraging the Monte Carlo Markov Chain (MCMC) policy algorithm. In [53], the authors used Bayesian Inference to predict the intention of an agent navigating in a partially occluded environment. Intention is represented by the end-point goal that the agent is trying to reach and is assumed that the agent will most likely take the shortest path to the goal, with some uncertainty in its transition model.

2.0.2 Safety Critical Wireless Networks

Channel quality prediction is a well-studied area in Wireless Networks such as Wireless Sensor Networks (WSN), or mobile/vehicular ad-hoc networks due to the lossy and dynamic behavior of the wireless links. A large group of research works that study the empirical characterization of wireless networks show that radio links are often fluctuating over time [54, 55] or space [56, 57] mainly due to environment and multipath propagation, or the interference which results from concurrent transmissions in a wireless network.

To estimate the link quality, many works combine multiple variables extracted from the network physical and link layers to form a more comprehensive and robust metric to quantify the link quality [58, 59]. For example, a basic idea is to observe the packet reception during a certain period of time, then predict the expected packet delivery rate (PDR) in the future from the past observation. Baccour et al. [60] classifies the link quality metrics into hardware and software based indicators. The hardware metrics are the physical (PHY) layer parameters of the signal such as the reception signal strengh intensity (RSSI) [61], the signal to noise ratio (SNR) , and the 802.11n channel state information (CSI) [62]. On the other hand, the software metrics are computed from higher level network information such as the expected count transmission (ETX) [63], or packet reception rate [64]. Our proposed method leverages the benefit of both groups of metrics by modeling the correlation between hardware (CSI) and software (PDR) metrics and combine them with the physical environment characteristics.

The majority of metric-based methods combine the recent and old measurement of the signal parameters (e.i. packet delivery rate) by calculating the Exponentially Weighted Moving Average (EWMA) [65] or Window Mean Exponential Weighted Moving Average (WMEWMA) [66]. However, due to dynamic nature of the wireless communication, the simple observation of a metric is not enough for link quality estimation [60]. Another group of works use pattern matching to predict the future behavior of a link. For example, Farkas et

al. [67] records a time series of the SNR and find the best match in the historical data to estimate future behavior of the SNR. However, link Quality prediction is even more challenging in connected vehicle use case due to fast channel fluctuations. So, our proposed method leverages past and current measurements of the channel and physical information, and combines them with future trajectory in a deep learning model. Therefore, it extracts temporal and spatial characteristics of the channel from short-term and long-term historical data, thus achieving a higher accuracy.

The historical data used for link quality prediction can be monitored actively or passively. In active mode, a node monitors the link quality by sending probe packets such as broadcasting beacon messages [63, 68, 69]. However, this approach is considered as costly due to the communication overhead. Unlike active link monitoring, passive mode exploits existing traffic without adding any communication overhead [54, 70, 71]. However, it may lead to the lack of up-to-date link measurements for lower data rates. Our proposed prediction method can be used with active or passive modeling based on the use case and trade-offs of the system.

Despite rate adaption or hand-off applications in wireless communication in which packet delivery rate can be directly applied, in the use case of connected vehicle communication, the vehicle controller cannot use PDR because it is an averaged value and can result in random communication delay in practice. Therefore, safety cannot be guaranteed. To fill this gap, we leverage *probabilistic model checking*. Through checking all possible system executions, probabilistic model checking can provide best/worst-case performance bounds for a wireless protocol. *PRISM* [72] is a probabilistic model checker which has been widely used in probabilistic model checking for wireless protocols, and Kwiatkowska et al. [73] gave an overview about different types of probabilistic temporal-logic properties of wireless network protocols that can be analyzed using this probabilistic model checker.

There have been many recent developments for the synthesis and formal analysis of control and communication schemes for networked, multi-agent mobile systems. For example, one approach is to plan the agent mobility and timing of transmissions for multiple coordinating agents to guarantee maintenance of a connected communication graph [74, 75], with mechanisms to recover from loss of connectivity [74]. To improve scalability, other approaches define “barrier certificates" (criteria) to certify that a single agent will satisfy certain safety or communication requirements [76, 77], and these barriers can be composed into a single, valid certificate for the entire multi-agent system. Prior work has also demonstrated the ability to produce connectivity guarantees by synthesizing control strategies from formal specifications [19] or using correct-by-construction control approaches [17, 18, 78, 79].

However, these techniques tend to assume a stationary communication channel and/or reason about communication requirements as distance-based connectivity requirements. Our work enhances these control synthesis and formal analysis methods by providing techniques that allow agents to reason about wireless communication not just as a function of distance, but as a function of the agents' context in a complex, dynamic physical environment.

In addition, there is a vast literature on vehicle platooning and collaborative control [80–84]. These works generally consider the longitudinal control of multiple vehicles under uncertain communication. The main differences with our work are two-fold: the use of both trajectory prediction and communication prediction in concert. To our knowledge, the use of nonlinear model-predictive control to explicitly reduce communication latency–while simultaneously assuring safety and improving traditional performance metrics–is novel.

2.0.3 RL Safety

Since Deep RL lacks inherent safety guarantees, a considerable amount of recent work has focused on designing new RL algorithms that expressly incorporate safety considerations. The literature on safe RL can be classified according to three broad approaches. The first approach focuses on modifying the training algorithm to take into account safety constraints. Representative examples of this approach include reward-shaping [85], Bayesian and robust regression [86–88], and policy optimization with constraints [89–92]. Unfortunately, this approach does not provide provable guarantees on the safety of the trained controller. The second approach focuses on using ideas from control theory to augment the RL agent and thereby provide safety guarantees. Examples of this approach include the use of Lyapunov methods [93–95], safe model predictive control [96], reachability analysis [97–99], barrier certificates [100, 100–106], and online learning of uncertainties [107]. Unfortunately, methods of this type suffer from being computationally expensive, specific to certain controller structures or else employ training algorithms that require certain assumptions on the system model. Finally, the third approach focuses on applying formal verification techniques (e.g., model checking) to verify the formal safety properties of pretrained RL agents. Representative examples of this approach are the use of SMT-like solvers [108–110] and hybrid-system verification [111–113]. However, these techniques only assess the safety of a given RL agent rather than design or train a safe agent.

Chapter 3

Vulnerable Autonomous Systems Safety

3.1 Online Controller Adaptation for Secure Autonomous Vehicles

In this section, we are interested in finding a policy to adapt online the low-level control law of an autonomous vehicle subject to time varying delays to guarantee safety constraints (e.g., avoid collisions with obstacles, entering undesired regions in the environment).

3.1.1 Problem Formulation

We assume that the robot dynamics can be represented as a continuous linear time invariant (LTI) system of the following form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) \end{aligned} \tag{3.1}$$

where $\mathbf{x}(t)$, $\mathbf{u}(t)$, and $\mathbf{y}(t)$ are the state vector, control vector, and sensor measurements vector at time t respectively. A, B, and C are the system matrix, the input matrix, and the output matrix respectively.

A typical autonomous driving control algorithm consists of a high-level and a low-level controller as depicted in Fig.**??**. The high-level (HL) controller is typically used for path planning to generate collision-free trajectories to goal locations while the low-level (LL) controller is used to track the set of points along the path computed by the HL controller. The low-level controller is usually implemented as a computer application which runs in a discrete fashion, and thus it needs to be described with a discrete time model as

follows:

$$\begin{aligned} \mathbf{x}(k+1) &= A_d\mathbf{x}(k) + B_d\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) \end{aligned} \tag{3.2}$$

where $A_d = e^{At_s}, B_d = \int_0^{t_s} e^{A\lambda}Bd\lambda$, and t_s is the sampling time used to discretize the system.

Protecting the controller application by applying cyber-security techniques (discussed in more depth in the next section) imposes performance overhead on the controller. This performance overhead can be time varying, depending on the implementation and the complexity of the controller code.

From a controller point of view, adding a delay $\delta(k)$ at each iteration in a control loop that is running with sampling time t_s, is equivalent to a control loop that runs at $t_s + \delta(k)$ rate. Therefore, the discrete model of the system, considering time varying delays becomes:

$$\begin{aligned} \mathbf{x}(k+1) &= A'_d(k)\mathbf{x}(k) + B'_d(k)\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) \end{aligned} \tag{3.3}$$

where

$$\begin{aligned} A'_d(k) &= e^{A(t_s+\delta(k))} \\ B'_d(k) &= \int_0^{t_s+\delta(k)} e^{A\lambda}Bd\lambda \end{aligned} \tag{3.4}$$

Designing a robust controller may take care of these delays, however if the overhead is large, this issue can create instabilities and possible performance degradation and unsafe behavior since the input may be applied for longer sampling intervals than the designed rate, driving the system to unexpected and unsafe states (i.e., the AV may run into an obstacle or enter a safety critical region). In this work we address this issue by computing a control policy that guarantees that at any time the AV is running safely. For the sake of simplicity, we consider obstacle avoidance case studies, however the proposed framework can be applied to other types of operations involving in general cyber-physical systems.

Formally, in this work we are interested in solving the following problem:

Problem 1 Online control adaptation for secure and safe AV operations under delayed control: *An autonomous vehicle (AV) is tasked to complete a mission over an obstacle populated environment $\mathcal{W} = \mathcal{F} \cup \mathcal{O}$ in which $\mathcal{F}$ represents the obstacle-free region of the environment and vice versa $\mathcal{O}$ is the region occupied by obstacles. The AV is modeled by (3.3) in which delays, δ, are due to security mechanisms running on the*

controller to protect the system against malicious cyber attacks. Given these constraints, the current state of the system x, the desired input u with no delay, and the maximum delay that we can expect δ^{max}, the objective is to find a control policy $\hat{u} = f(x, u, \delta^{max}, t)$ such that $\boldsymbol{x}(t) \notin \mathcal{O}, \forall t > 0$

In other words we are interested in finding a policy to guarantee that a secured AV is safely performing its objective.

3.1.2 Approach

The framework that we propose consists in a series of steps in which we first predict the future inputs and states of the system, and then replan these inputs to avoid the possibility to reach unsafe states. To achieve this behavior we consider both an estimate of the current delay δ^e and an upper bound on the maximum possible delay δ^{max}. Because $\delta(k)$ is not known a priori, a conservative approach would be to compute u based on δ^{max}. The generated input will drive the AV always to a safe state, however the performance of the system will be greatly reduced because the system will run unnecessarily slow. To maintain system's performance we propose the following adaptive procedure in which the input is selected considering the current state of the system in relation to the unsafe states to avoid.

The first step in our approach uses the Model Predictive Control (MPC) [114], to predict the evolution of the AV states over a finite horizon h.

$$\begin{aligned} J(\mathbf{x}(k), \mathbf{u}(k)) = & \min_{\mathbf{u}(k)} \quad \sum_{k=0}^{h-1} e_x^T(k) Q e_x(k) + e_u^T(k) R e_u(k) \\ & \text{subject to} \quad \mathbf{x}(k+1) = A'_d(k)\mathbf{x}(k) + B'_d(k)\mathbf{u}(k) \end{aligned} \tag{3.5}$$

The result of (3.5) is a series of inputs $[\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(k), \ldots \mathbf{u}(h)]$ from the current time to the h time horizon. $e_x(k)$ and $e_u(k)$ are the errors between the current state x_c and the desired state x_d and the error between consecutive inputs, respectively. Q and R are the weight matrices and A'_d and B'_d are calculated according to (3.3). To determine the correct input to apply to our system, one approach is to solve the MPC optimization problem at each time step k by changing A'_d and B'_d according to the measured delay $\delta(k-1)$. However, this approach cannot be realized because the runtime overhead introduced by the cyber-security techniques is time varying and not known a priori before implementing the control law. Instead in this work, we consider an exponential weighted moving average algorithm (EWMA) that compute an estimated value of

the delay at time k based on the previous estimation and the previous measured delay as follows:

$$\delta^e(k) = (1-\alpha)\delta^e(k-1) + \alpha\delta(k-1) \tag{3.6}$$

where $\delta^e(k)$ is the estimated delay at time step k, $\delta(k-1)$ is the measured delay at time step $k-1$, and α is a weighting parameter. We can set an operating envelope to determine the maximum delay δ^{max} which can occur at any time step. This operating envelope can be obtained using a variety of worst-case execution time (WCET) and profiling techniques as outlined in [115].

Once we have an estimate of the delay δ^e, we solve the following MPC optimization problem at each time step k.

$$\begin{aligned} A'_d(k) &= e^{A(t_s+\delta^e(k))} \\ B'_d(k) &= \int_0^{t_s+\delta^e(k)} e^{A\lambda} B d\lambda \end{aligned} \tag{3.7}$$

The control input $u^e(k)$ that we obtain by solving (3.7) is then used to predict the state $x^e(k+1)$ covered by the AV in one time step. By applying $u_e(k)$, one of the following three cases can occur:

1. $0 < \delta(k) < \delta^e(k)$: The actual delay is less than the expected delay, so the system will move to the next state but with sub-optimal performance than the one computed by the MPC.

2. $\delta(k) = \delta^e(k)$: The actual delay equals the estimated delay, so the system will move to the next state with the optimal performance computed by the MPC.

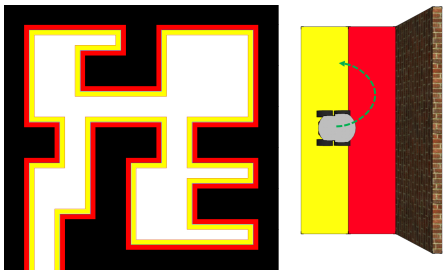
3. $\delta^e(k) < \delta(k) < \delta^{max}$: The actual delay is greater than the estimated delay, which means that the system states may evolve to undesired and unsafe states.

Thus, we propose an algorithm that adapts the control input to guarantee the safety of the AV and at the same time minimize the degradation in its performance.

Given the shape, kinematics of the AV, and δ^{max} we can inflate the unsafe regions or obstacles to construct an inflated obstacle set $\mathcal{S}$ that satisfies the following condition:

$$\forall \mathbf{x}(k) \subset \mathcal{S}, \exists u^c \in U \text{ such that } \mathbf{x}(k+1) \not\subset \mathcal{S} \tag{3.8}$$

where $\mathbf{x}(k+1) = A_d'^{max}\mathbf{x}(k) + B_d'^{max}u^c, U = \{u | u_{min} \leq u \leq u_{max}\}$, with u_{min} and u_{max} the minimum and maximum controller inputs, respectively.



(a) (b)

Figure 3.1: Obstacles inflation. (a) Configuration space after inflating obstacles, (b)The red regions are constructed according to AV's shape and turning radius

Fig. 3.1(a) shows an example of inflated obstacles. The figure shows a 2-D view of the configuration space of an AV. The yellow shaded regions indicate the points inside the set $\mathcal{S}$. The width D of the yellow region is calculated as $D = v^{max}(Ts + \delta^{max})$, where v^{max} is the maximum speed of the AV. Therefore, an AV that lies inside a safe region cannot cross this region in one single controller time step even if maximum delay occurs. The width of the red shaded regions is calculated based on the shape and the turning radius of the AV to guarantee that it is able to apply a steering input that drives it outside the unsafe region without hitting an obstacle, Fig. 3.1(b).

By constructing the set $\mathcal{S}$, we can develop an adaptation algorithm that outputs a control input u^c only if the AV is inside the inflated obstacles set $\mathcal{S}$. To compute u^c, we add $\mathbf{x}(k+1) \not\subset \mathcal{S}$ as a constraint in the MPC optimization problem. This is captured by $MPC_{Cons}(\delta^{max})$ inside the algorithm to denote the MPC controller having this extra constraint.

3.1.3 Simulations & Experiments

In this section, we present simulations and experimental results to evaluate the proposed online controller adaptation algorithm.

UGV Model

In this work we consider that the dynamics of the AV can be described by the following non-holonomic differential drive skid steering model that reflects the dynamics of our ground robot used during experiments:

$$\begin{aligned} \dot{x} &= \frac{R}{2}(v_r + v_l)\cos\theta \\ \dot{y} &= \frac{R}{2}(v_r + v_l)\sin\theta \\ \dot{\theta} &= \frac{R}{2}(v_r - v_l) \end{aligned} \tag{3.9}$$

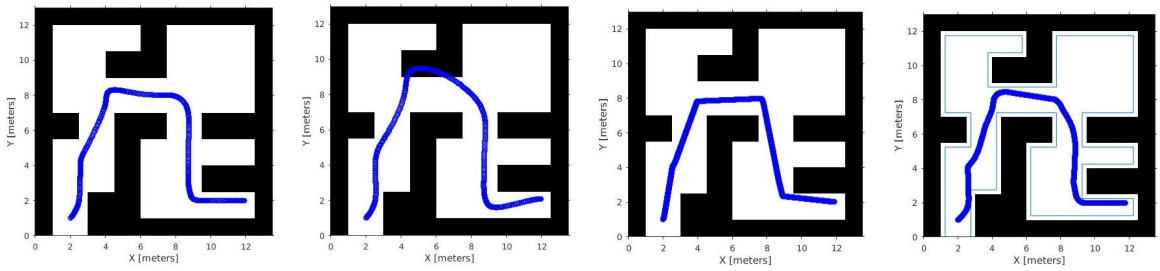
where $\dot{x}$ and $\dot{y}$ are the $x-y$ velocity components of the vehicle. v_r and v_l are the linear velocity of the right and left side wheels of the robot, respectively, and R is the radius of the wheel. θ is the angle of the vehicle with respect to a global frame and $\dot{\theta}$ is its angular velocity.

Simulations

In order to show the effect of variable overhead delays applied to a low-level controller of an AV, we consider a scenario in which an unmanned ground vehicle (UGV) navigates in a cluttered environment toward a desired goal. The task of the UGV is to go from a starting position of coordinate (2,1)m to a goal located at (12,2)m. We use a *Probabilistic Roadmap algorithm* (RPM) implemented inside the Matlab Robotics toolbox to generate a collision-free path from the start to the end points. This path consists of tuples of waypoints. The task of the low-level controller is to visit all the waypoints along the path until it reaches the goal. The controller outputs the acceleration of the vehicle in x and y directions. The controller is designed to run with a sampling time $t_s = 0.01$s.

Fig. 3.2(a) shows the scenario in which the vehicle completes the task successfully in 6.3s without incorporating any delays in the controller. The effect of cyber security protection techniques is simulated in Fig. 3.2(b-d) by imposing a cyclic overhead delay that varies between 0 and 10x overhead delay. We use this specific overhead function because it has a similar behavior of the CPU utilization for virus scanning described in section **??**. Without adapting the controller, the vehicle starts to drift away from the collision-free path and runs into obstacles as shown in Fig. 3.2(b). In Fig. 3.2(c) we show the case in which a conservative controller is used assuming always maximum delay. The vehicle remains inside the safe regions at all times however the simulation ends up running slower taking 43.4s. Finally, in Fig. 3.2(d) we show the case in which the online adaptation algorithm is operating considering $\alpha = 0.7$. The vehicle is able to reach the goal

without hitting obstacles in 24.7s. Table 3.1 summarizes and compare the different execution times obtained in this simulation scenarios.



(a) (b) (c) (d)

Figure 3.2: Simulation Results. (a) No Overhead Delay (b) No adaptation (c) Conservative navigation (d) Adaptive navigation

Table 3.1: Comparison between navigation task execution times in different scenarios.

Scenario	**Task Execution Time(s)**
No overhead	6.3
Overhead + conservative controller	43.4
Overhead + online adaptation	18.1

Experiments

The experiments were run on a Clearpath Jackal UGV [116] equipped with a core-i7 CPU, lidar, camera, GPS, IMU, and wheels encoders. Experiments were performed inside our Vicon Motion Capture arena that allows precise tracking of the position and the orientation of the UGV. Our motion planning control algorithms were developed under the Robot Operating System (ROS) framework which consists of x86 applications running on top of a Linux operating system.

Integrating software level cyber security techniques on a real autonomous vehicle

The first experimental result that we present is the implementation of a N-variant system, called *Double Helix* [1]. Double Helix is used to protect our ROS nodes for mapping and navigation. The goal of this experiment is to provide an example of how we can protect a controller software implemented on a real robot and evaluate its performance in terms of overhead. Fig. 3.3 summarizes the process that we followed to implement Double Helix on our robot. The robot is tasked to go from an initial position to a goal while avoiding an obstacle along its path. Double Helix generates different variants of the ROS source nodes, and finally, each variant is deployed on the Jackal. Each variant is protected with different techniques picked from

Table 3.2: Average runtime overheads for two ROS nodes protected by Double Helix [1]

Type of Overhead	*slam_gmapping*	*move_base*
Overhead in CPU load	17.8%	12.5%
Overhead in Memory consumption	3.0%	2.56%

the Diversity Transformation Library in Double Helix. In each run (i.e., for every variant) we recorded the performance overhead imposed by the protected controllers nodes (*slam_gmapping* and *move_base* nodes) measuring delays up to 30%. Table 3.2 shows the average performance overhead imposed by the security techniques on the two nodes used for navigation operations.

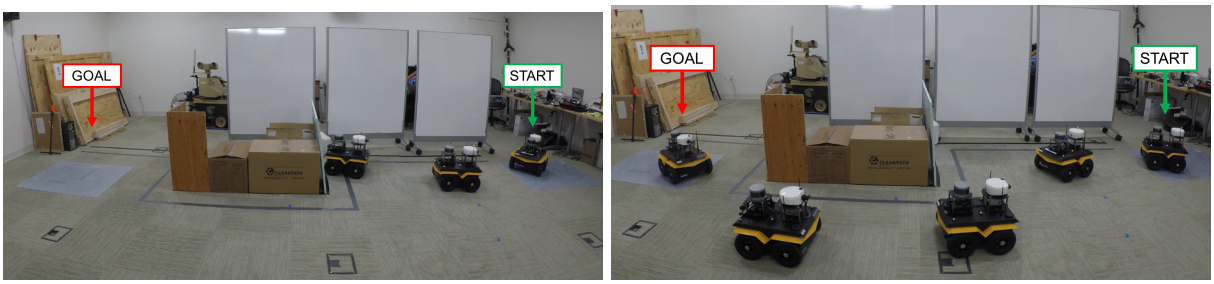
Figure 3.3: Generating and deploying Double Helix variants into our ROS based UGV

Online Adaptation Control with Unknown Overhead

In this section we show the results obtained by running our online adaptation algorithm when large random delays are applied to our controller nodes. Note that here we introduce large artificial delays into the controllers, instead of implementing directly the techniques introduced in the previous section, because the overall overhead effects are more visible by inflating the delays to 10x, typical of other cyber security techniques. Specifically here we study the effects of these delays on our platform by running several trials with increasingly higher delays until the system starts to misbehave (i.e., it hits the obstacle).

Without any delay, the UGV accomplishes its task in 34s. When we increase the overhead delay above 10x the UGV hits the obstacle, as shown in Fig. 3.4(a).

When we apply the conservative controller designed for δ_{max}, The UGV completes its task in 50s. Finally, in Fig. 3.4(b) we show a snapshot of the same operation running the online adaptation algorithm with a square wave cyclic runtime overhead pattern with a maximum delay of 10x. The UGV was able to navigate safely avoiding the obstacle in 42s.



(a) (b)

Figure 3.4: Experimental Results. A sequence of snapshots for (a) a UGV hitting obstacles due to improper input with large delay, (b) a UGV avoiding obstacles while applying the online adaptation algorithm presented in this work.

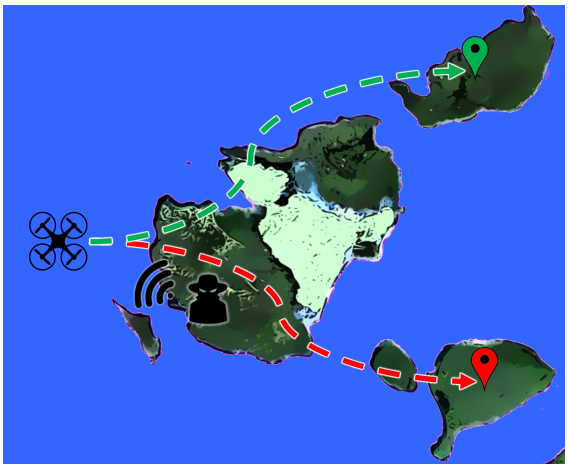


Figure 3.5: Pictorial representation of the situation envisioned in this work in which a malicious attacker spoofs a sensor like a GPS on a UAV to hijack it to an undesired destination (red goal) from the desired route (green trajectory)

3.2 Drone Hijack Attacker Intention Prediction

In this work, we are interested in addressing the problem illustrated in Fig. 3.5. We consider the case where an autonomous vehicle, equipped with multiple sensors, is tasked to perform a go-to-goal navigation. A malicious attacker performs a coordinated attack by spoofing one or more of the sensors on the vehicle. The goal of the attacker is to hijack the vehicle to an undesired goal (not known beforehand) while hiding within the sensor and actuator noise profile and disturbance model of the system. Our goal is to develop a technique to infer the intention of the attacker and recover the system before reaching the undesired state. To this end, we leverage sensor redundancy and the theory of Inverse Reinforcement Learning to: i) predict the intention of the attacker ii) determine the set of compromised sensors and iii) recover the uncompromised state of the vehicle to continue its desired objective.

3.2.1 Problem Formulation

We consider an autonomous vehicle that performs a navigation task to a desired goal $g^* \in \mathcal{G}$ in a stochastic environment. The vehicle is equipped with N sensors. Sensor readings are fused together to estimate the state of the vehicle. We also consider a malicious attacker that can spoof a subset of sensors $\mathcal{S}^a \subset \mathcal{S}$. The goal of the attacker is to drive the vehicle towards a different undesired location $g^a \neq g^*$ while hiding its attack vector within the noise profile of the spoofed sensors and the disturbance model of the stochastic environment. Fig. 3.6 summarizes the situation in which a sensor spoofing attack stealthily begins at t_a. The attack remains stealthy until t_d, when the noise margins of the spoofed sensors don't overlap anymore with the noise margins of the uncompromised sensor. At t_d, an attack can be detected due to the differences between sensor readings. However, at that point, it is not obvious how to distinguish between the compromised and uncompromoised sensors, since all sensor readings comply with the uncertainty and disturbance model of the vehicle dynamics and the environment. Hence, here we are interested in solving the problem of inferring the intention of an attack by observing and comparing measurements data and control inputs, recognizing which sensors are compromised and finally recover the system.

3.2.2 Approach

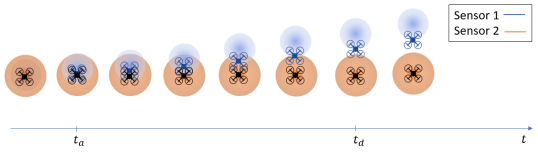


Figure 3.6: An attack on sensor 1 (blue) starts at t_a while sensor 2 (orange) remains uncompromised. The attack can be detected at t_d since the two sensor readings don't overlap anymore.

Specifically, we consider that the autonomous vehicle follows an optimal mission policy π^* to perform its navigation task to the desired goal.

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} Q^{\pi^*}(s, a, R) \tag{3.10}$$

An attack is launched at a previously unknown time t_a acting according to a policy π^a described as follows:

$$\pi^a(s) = \underset{a \in A}{\operatorname{argmax}} Q^{\pi^a}(s, a, R) \tag{3.11}$$

where R has the following form for both mission and attack policy:

$$R(s) = \begin{cases} C & s = g^* \ \vee s = g^a \\ -\epsilon & s \neq g^* \neq g^a \\ -C & s \neq g^* \end{cases} \tag{3.12}$$

with $C \in \mathbb{N}^+$ and $\epsilon << C \in \mathbb{N}^+$. The term ϵ represents the cost to take any action.

To summarize, the vehicle uses fused data from all sensors and navigates to g^* according to an optimal policy π^*. An attacker spoofs sensors data such that the fused sensor information indicates that the vehicle is in a different state making the vehicle take an action that follows the attack optimal policy π^a leading it towards g^a. The attacker is assumed to know both the optimal mission policy π^* and the MDP parameters of the system.

For the sake of simplicity, we consider that the sensors on board of the vehicles are providing an accurate state of the vehicle. For the specific case studies investigated in this work we consider position states.

We, formally, describe the problem as follows: Consider a robot equipped with N sensors, tasked to go to a goal $g^* \in \mathcal{G}$ following an optimal policy π^* obtained by solving an MDP. Assume that at an unknown time t_a, one or more sensors, up to $N-1$ are compromised by an attack whose intention is to steer the robot toward a goal $g^a \in \mathcal{G}$ with $g^a \neq g^*$. Given a super-set of observations $\mathcal{O}_{1:N,\Delta t} = \{\mathcal{O}_{1,\Delta t}, \mathcal{O}_{2,\Delta t}, ..., \mathcal{O}_{N,\Delta t}\}$, where $\mathcal{O}_{i,\Delta t} = \{(s_{i,t_d}, a_{t_d}), (s_{i,t_d+1}, a_{t_d+1}), ..., (s_{i,t_d+n}, a_{t_d+n})\}$ is the finite set of measurement-action pairs for each sensor $i = 1 \dots N$ recorded over the period $\Delta t = [t_d, t_d + n]$ with $n \in \mathbb{N}$ where the current time $t = t_d + n$, find a policy to predict the goal g^a of the attacker, determine the set of compromised sensors $\mathcal{S}^a$ and recover the vehicle by implementing the optimal policy π^* on the set of uncompromised sensors $\mathcal{S}^u = \mathcal{S} \setminus \mathcal{S}^a$.

we leverage the BIRL approach proposed in [51] where the inference of the reward function posterior $Pr(R|\mathcal{O})$ is a function of the reward function prior $Pr(R)$ and the likelihood of the observations $Pr(\mathcal{O}|R)$ as follows:

$$Pr(R|\mathcal{O}) = \frac{Pr(\mathcal{O}|R)Pr(R)}{Pr(\mathcal{O})} \tag{3.13}$$

where, $Pr(\mathcal{O})$ is the probability distribution of $\mathcal{O}$ over the entire space of reward function R. The normalization constant $Pr(\mathcal{O})$ is usually hard to compute as it requires performing multiple integral calculations which are usually not feasible to compute analytically.

Instead, we leverage the MCMC sampling algorithm presented in [117] to approximate the distribution of the posterior $Pr(\tilde{R}|\mathcal{O})$ without the need to calculate $Pr(\mathcal{O})$.

In order to approximate the posterior $Pr(R|\mathcal{O})$ using an MCMC algorithm, we need to compute two terms at each iteration: i) the prior $Pr(R)$ and ii) the likelihood of the observations $Pr(\mathcal{O}|R)$. For our problem, we apply the following MCMC iterative algorithm for each sensor i to calculate the mean of the attacker's goal posterior estimate and determine the subset of the compromised sensors. First, we draw a sample g_j from the prior distribution of the undesired goals set $\mathcal{G} \setminus \{g^*\}$. Then, we compute the likelihood of observing the set of sensor readings and actions taken during a period $t_d : t$ following an optimal policy of the agent to go to a potential goal g_j, $\forall g_j \in \mathcal{G}$ as follows:

$$Pr(\mathcal{O}_{i,t_d:t}|g = g_j) = \frac{e^{\beta \sum_{t_k=t_d}^{t} Pr(a_{t_k}|s_{i,t_k},g_j)}}{\sum_{b \in A} e^{\beta \sum_{t_k=t_d}^{t} Pr(b|s_{i,t_k},g_j)}} \tag{3.14}$$

where, β is a factor that indicates how confident we are that the agent is acting optimally. To calculate $Pr(a_t|s_{i,t}, g_j)$, we need to calculate the likelihood of taking the action a_t at state $s_{i,t}$ to go to goal g_j which is equivalent to calculating the action-value function (Q-function) of this action as follows:

$$Pr(a_t|s_{i,t}, g_j) = Q^*(s_{i,t}, a_t, R_j) \tag{3.15}$$

To obtain Q^*, we need to solve a standard MDP problem. Finally, we compute the posterior probability for each goal g_j (i.e., the probability that the goal is g_j) given the set of observations $\mathcal{O}_{i,t_d:t}$ as follows:

$$Pr(g_j|\mathcal{O}_{i,t_d:t}) \propto Pr(\mathcal{O}_{i,t_d:t}|g_j)Pr(g_j|\mathcal{O}_{i,t_d:t-1}) \tag{3.16}$$

The first term on the right-hand side (rhs) of the equation represents the likelihood of the observations given that the goal is g_j, which is computed using (3.14). The second term on the rhs represents the prior which is updated from the posterior calculated at the previous time $t-1$. After the MCMC iterations are completed, we evaluate the level of confidence that $g = g_j$ by calculating the mean of the posterior. The higher the value of the posterior mean μ_i, the higher the confidence we have in the estimation. Algorithm 1 summarizes the steps taken to perform the attacker's intention inference.

A recovery procedure is triggered if there exists a set of sensors that returns a state s_i such that the variance of the posterior probability $\nu_i \leq \tau$ where τ is a user-selected threshold. The higher the threshold the sooner a recovery will be initiated but a less precise inference will be computed. On the other hand, a small threshold may cause a delayed recovery. The sensors associated with such posterior probabilities are the compromised ones and thus are removed from considerations during recovery.

Once the set of the uncompromised sensors $\mathcal{S}^a$ is removed, the vehicle navigates back to the desired goal automatically according to π^*.

Algorithm 1: Attacker Intention Prediction

foreach *sensor* $i \in \mathcal{S}$ **do**
  while *iteration* $e < e_{max}$ **do**
    Sample a goal $g_j \in \mathcal{G} \setminus \{g^*\}$;
    $Pr(a_t|s_{i,t}, g_j) \leftarrow Q^*(s_{i,t}, a_t, R_j)$;
    $Pr(\mathcal{O}_{i,t_d:t}|g = g_j) \leftarrow$ from (3.14);
    $Pr(g = g_j|\mathcal{O}_{i,t_d:t}) \leftarrow$ from (3.16);
  end
  $\mu_i \leftarrow mean(Pr(g = g_j|\mathcal{O}_{i,t_d:t}))$;
  $\nu_i \leftarrow var(Pr(g = g_j|\mathcal{O}_{i,t_d:t}))$;
  if $\nu_i < \tau$ **then**
    $g_a \leftarrow \mu_i$;
    $i \in \mathcal{S}^a$;
  end
end

Active Exploration Inference

The time that the inference algorithm takes to converge depends on the states that the vehicle visits. If the majority of the visited states are associated with mission policy's actions that are similar to the attacker policy's actions, the inferred posterior may not reach the desired threshold value. In this case, we consider these states as *insensitive states*. On the other hand, the set of observations that contains states in which there is a discrepancy between the mission policy actions and the attacker policy actions, will lead to a faster convergence. In that case, we name these states as *sensitive states*. It is desired to visit sensitive states as soon as an attack is detected.

To solve this problem, we propose an *active exploration* policy in which the vehicle perturbs its motion from the optimal policy to visit more sensitive states. This idea is reminiscent of reinforcement learning in which the environment is explored to learn the optimal policy. The idea is to perform a local search in the neighbor states at each time step to choose the action that will maximize the discrepancy in mission policy and all the weighted estimates of attacker's polices. *Active Exploration* works as follows. At each time step t, next action a^* is chosen by looking into each possible combination between the current sensor reading s_i, and next reachable states s_i'. The current action is selected by maximizing the probability of obtaining an action different from the optimal action in the next step. A discrepancy factor W_a for each $\{action, next\, state, goal\}$

tuple is calculated based on the transition probability T to reach next state s'_i from s_i as follows:

$$W_a \leftarrow W_a + \nu_i T(s_i, a, s'_i) \tag{3.17}$$

We only update W_a if $\pi^{\mu_i}(s'_i) \neq \pi^*(s'_i)$, where π^{μ_i} is the policy used to reach the estimated attacker's goal at the current time step. The posterior estimate μ_i, and the variance ν_i are calculated as in Algorithm 1. Finally, we pick the action to apply, a^*, according to:

$$a^* = \underset{a \in A}{\operatorname{argmax}} \mathcal{W} \tag{3.18}$$

where, $\mathcal{W} = \{W_1, W_2, ..., W_{|A|}\}$ is the set of weights for all the possible actions. We apply a^* at each time step with a probability $\delta, 0 < \delta < 1$ that decreases over time. As before, also this procedure will terminate once $\nu_i \leq \tau_i$ and a recovery process will start at that time.

$$Pr(a_{i,t}|s_{i,t}, g_j) \propto e^{-\beta \|a_t - a_{cl}\|_2} \tag{3.19}$$

3.2.3 Simulation Results

In this section, we present simulation results for a UAV navigation case study in a stochastic environment.

The goal of these simulations is to show the following behaviors:

- An attacker can hijack the vehicle to an undesired destination by spoofing its sensors while hiding the attack vectors within the noise profiles of the sensors and the UAV model uncertainties.
- Our technique for intent inference can infer the intention of the attacker and the set of compromised sensors.
- The Active Exploration approach can improve the intention prediction.

Simulation Setup

In the simulations that follow, we consider a 10×10 square cells environment with each cell having 1m side length. $R(s) = 100$ for the desired goal, $R(s) = -100$ on undesired goals located on the perimeter of the environment and $R(s) = -3$ on the remaining cells. In normal operation conditions (i.e., with no attack), the UAV takes an action at each cell according to its mission optimal policy π^* in (3.10). The set of actions A that the UAV can take at any cell is defined as follows: $A = \{$*move forwards {F}, move backwards {B}, move*

right {R}, move left {L}}. The UAV has two noisy sensors measuring the position state. State estimation is performed using a Kalman filter. Noise and uncertainties associated with the motion of the UAV are captured through transition probabilities as follows: for any possible action, there is a 0.8 probability to reach the desired state and 0.1 probability to reach each of the adjacent cells in the environment. Fig. 3.7 illustrates the situation.

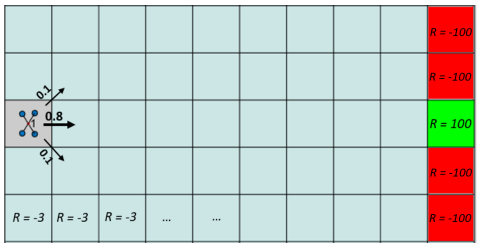
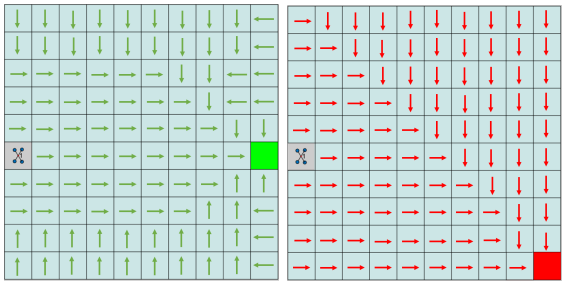


Figure 3.7: The discretized environment used during simulations with the reward and transition probability models.

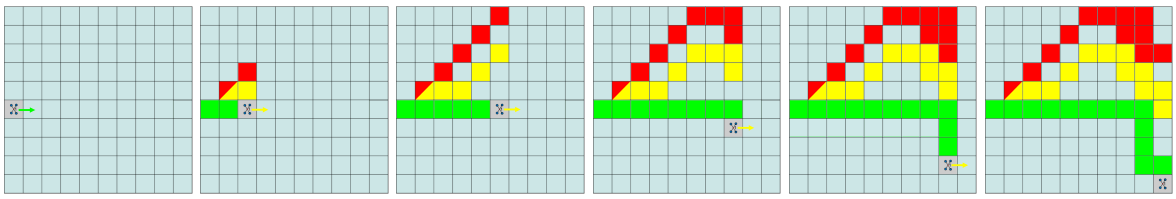
Sensor Spoofing Attack

In this section, we demonstrate the case in which an attacker is spoofing one of the two sensors on the UAV an no protection mechanisms are deployed. We assume that the attacker has full knowledge of the system



(a) (b)

Figure 3.8: (a) Optimal mission policy. (b) Optimal attacker's policy.



(a) (b) (c) (d) (e) (f)

Figure 3.9: Simulation of a UAV navigation case study under attack with no resiliency mechanisms. Red colored cells represent the readings from the compromised sensor, green cells show the uncompromised sensor readings while yellow cells represent the estimated states fusing both sensors. In (a) the UAV is not under attack. In (b-f) the attack is applied hiding within the system uncertainty.The UAV eventually reaches the undesired destination in (f).

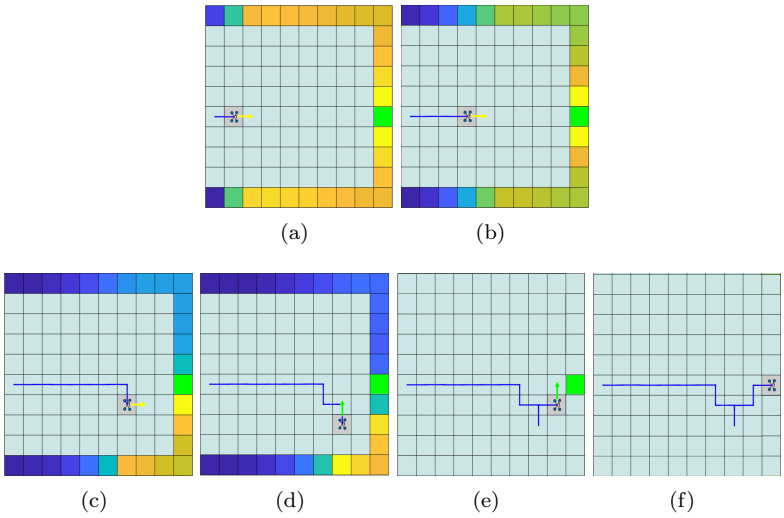


Figure 3.10: Simulation of intent inference with the same attack depicted in Fig .3.9. The gradient color map in (a-d) shows the likelihood of each possible goal on the perimeter of the environment with blue=low probability and yellow=high probability. In (d) the inference confidence level exceeds a given threshold, the compromised sensor is discarded, and the UAV is recovered from the attack in (e-f).

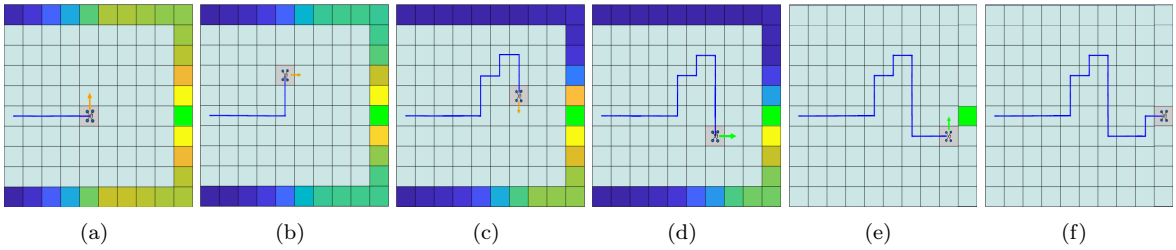


Figure 3.11: Simulation of intent inference with active exploration. In (b) active exploration is activated driving the system sub-optimally to infer more the attack intention and before reaching regions in the environment too close to the undesired state. Differently from the previous simulation the intention is estimated farther away from the undesired goal and as before in (e-f) we show the recovery operation while discarding the compromised sensor readings.
and exploits the uncertainty in the system model to hide the attack vectors. The goal of the attack is to hijack the UAV towards a different goal from the desired one. In other words, the attacker wants the UAV to follow a different policy π^a as depicted in Fig. 3.8 where the arrows on each cell represent the optimal actions associated with each policy.

In Fig. 3.9, we show a sequence of snapshots in which an attacker spoofs one of the two sensors. Based on the state estimate obtained by fusing the compromised measurement with the uncompromised measurement, the attacker is able to hijack the UAV to the undesired state.

Attacker Intention Inference

In the simulation shown in Fig. 3.10, we deploy our intent inference approach. Here, we apply the same attack illustrated in the previous section. The border cells of the grid (with exception of the desired goal) represent possible attacker's goal locations. We assume that the UAV doesn't have any prior knowledge about the attacker's intention and thus the prior of the attacker's goal is drawn initially from a uniform distribution. The color gradient on the sequence of snapshots in Fig. 3.10 shows the convergence of the posterior estimate toward the region where the attacker's goal is located. In Fig. 3.10(d) the variance of the estimated mean of the posterior of the attacker's goal goes below a threshold value. At that point, the compromised sensor is identified and discarded from the state estimation and the system is driven back toward the desired goal 3.10(e-f).

Active Exploration

In the simulation depicted in Fig. 3.11, we show the benefit that we gain by applying the *active exploration* technique. Differently from Fig. 3.10, in Fig. 3.11, the UAV doesn't take always the optimal action from π^*. Instead, it computes the controller action using (3.18). The UAV explores more sensitive states in the environment before getting too close to the attacker's goal. Although the UAV ends up taking a longer path, this approach prevent reaching states too close to the undesired regions. Before switching to recovery, the estimated posterior mean value for the undesired goal for the active exploration case was recorded $\mu = 0.0319$ against $\mu = 0.0239$ calculated in the same state by using inference without exploration, hence, showing that the active exploration was also able to increase the level of confidence about the inference.

Chapter 4

Wirelessly Connected Autonomous Systems Safety

In this chapter, we consider a system of vehicles in a platoon in which there is a leading vehicle and several following vehicles, and each vehicle should keep a safe distance with its preceding vehicle. We assume that all of the vehicles move in a single lane and the platoon system is decentralized. The decentralized platoon system means that each vehicle generates its own motion policy based on the data received from the preceding vehicle through an established communication channel. Imagine that there are $n+1$ vehicles in the platoon and their states are represented by $\mathbf{x}_i(t) = [x_i(t), y_i(t), \psi_i(t), v_i(t)]', \forall i = \{0, ..., n\}$, in which the zero index refers to the leading vehicle and $x_i(t)$ and $y_i(t)$ show the location of central point of the i^{th} vehicle at time t. Also, $\psi_i(t)$ and $v_i(t)$ are heading angle and velocity of the vehicle. The euclidean distance between the $(i-1)^{th}$ and i^{th} vehicles at time t is given as $d_i(t), \forall i = \{1, ..., n\}$. To guarantee the safety and provide a collision-free trajectory for each vehicles we take into account a *Time To Collision* factor which is denoted by $\gamma_i(t), \forall i = \{1, ..., n\}$ and is defined as the time required for two vehicles to collide if they continue at their current speed and on the same path:

$$\gamma_i(t) = \begin{cases} \infty & v_{i-1}(t) \geqslant v_i(t) \\ \frac{d_i(t)}{v_i(t) - v_{i-1}(t)} & v_{i-1}(t) < v_i(t) \end{cases} \tag{4.1}$$

The control input vector consists of $a_i(t)$ and $\delta_i(t)$, which are acceleration and steering angle respectively, and is denoted by $\mathbf{u}_i(t) = [a_i(t), \delta_i(t)]'$. We assume that all of the vehicles in the platoon have a similar

dynamical system described by nonlinear continuous time equations [118] as follows

$$\dot{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i) \tag{4.2}$$

$$\mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i) = \begin{bmatrix} v_i cos(\psi_i + \beta_i) \\ v_i sin(\psi_i + \beta_i) \\ \frac{v_i}{l_r} sin(\beta_i) \\ a_i \end{bmatrix} \tag{4.3}$$

where β_i is the angle of velocity vector respect to the longitudinal axis of the car

$$\beta_i = tan^{-1}\Big(\frac{l_r}{l_r + l_f} tan(\delta_i)\Big) \tag{4.4}$$

and l_r and l_f denote the distance of the center of mass from the rear and front of vehicle.

To model the wireless channel, we consider Channel State Information (CSI) as a physical-layer metric to predict Packet Delivery Rate (PDR) as a link-layer metric. In a wireless link, the radio wave is transmitted from an antenna that propagates through the wireless channel, i.e. the environment. The dominant source of attenuation in a wireless medium is the diffusion of energy through the environment known as path loss, which is captured by the Friss transmission equation

$$P_r \propto \frac{P_t}{d^n} \tag{4.5}$$

where d is the distance between transmitter and receiver, p_r is receive signal power, p_t is transmit signal power, and n is the path loss exponent. The path loss exponent varies in different environments, which can be explained as the total of many complex effects caused by the reflection of radio waves from objects in the environment (known as shadowing effect). Besides path loss, multipath propagation is the most important effect on a wireless channel. In indoor and urban environments, RF signals bounce off objects such as metal and glass surfaces, which results in many copies of the signal arriving at the receiver along multiple paths. Therefore, the received signal is the superposition of all these paths, which could result in constructive or destructive summation.

CSI essentially models the channel gain coefficient (amplitude and phase) for each frequency channel, so it is a collection of $P \times Q$ matrices, which describes the RF paths between all pairs of P transmit and Q receive antennas for one frequency channel. For wireless channels with multiple paths, the CSI value from

the ith transmit antenna to the jth receive antenna at the kth frequency channel is

$$H_{ijk} = \sum_{n=1}^{N} \alpha_n e^{-j2\pi d_{ijn} c/f_k} \tag{4.6}$$

where α_n is the attenuation along the n^{th} path, d_{ijn} is the distance between the i^{th} transmit and the j^{th} receive antenna along the nth path, f_k is the frequency of the k^{th} subchannel, N is the number of paths, and c is the speed of light. Signal to Noise Ratio (SNR) relates with amplitude α_n as follows: $SNR = 10 \log 10(\alpha_n/\mathcal{N})$, where $\mathcal{N}$ denotes the average power of white noise. As can be seen in the equation 4.6, the phase component of CSI varies over both frequency and space; therefore, the wireless channel quality may alter from good to bad with a small change in path lengths or frequency of the signals. CSI is a fine-grained measurement of this effect and can capture the corresponding frequency selective fading and the effect of independent spatial paths. Deep fading in the channel response exhibits destructive interference of paths (i.e. paths cancel each other) and result in loss of information and accordingly loss of packets in the link-layer. Therefore, there is a direct relationship between physical-layer channel response and packet delivery rate (PDR) in the link-layer defined as

$$PDR = \frac{\#\text{ of received packets}}{\#\text{ of transmitted packets}}. \tag{4.7}$$

4.1 Problem Formulation

We make the following assumptions and addresses three core problems:

Problem 2 *Wireless Channel Prediction: Given the scenario described above, assume that the vehicles are the only moving objects in the environment. Assume that at each time $t = t_0$, each vehicle i has accurate estimates of its current state and the prediction of its future states over a time horizon $\boldsymbol{x}_i(t), \forall t \in [t_0, t_0 + T]$. Assume that every leading vehicle $i-1$ sends its current and future state estimates periodically to the following vehicle i with a transmission frequency f_t. The objective is, at any time t, predict the packet delivery rate $PDR_i(t)$ at each follower vehicle estimated over a future time horizon T.*

Problem 3 *Formal Analysis of Wireless Protocol: Given the assumptions in Problem 2, and given the wireless channel properties (e.g. packet delivery rate, clear channel assessment) at each follower vehicle, the objective is to compute the upper and lower latency bounds of the wireless transmission. The analysis shall be flexible enough to provide different levels of guarantees, for example that a packet with arrive in T seconds*

with likelihood $P > P_B$, where P_B can be any value $\in [0, 1]$. Practically, P_B can take on values such as 0.9, 0.99, or even greater reliability.

Problem 4 *Motion Planning and Control: Given the trajectory of the preceding vehicle $\boldsymbol{x}_{i-1}(t), \forall t \in [t_0, t_0 + T]$, find a control policy for the i^{th} vehicle $\boldsymbol{u}_{i-1}(t), \forall t \in [t_0, t_0 + T]$ that optimizes its performances in terms of amount of fuel consumption and traffic throughput while assuring safety. Because maintaining the predicted wireless channel at an acceptable level plays an important role in guaranteeing the safety and optimality of the trajectory, the proposed methodology should result in the least possible latency bound for the given time horizon T. Note that, although latency bound changes by varying γ_i (see eq. 4.1), the effects of the surrounding environment on latency bounds may be larger and are highly nonlinear. Therefore, we assume there is no close-formed function of latency bound in the parameters of the control problem.*

4.2 Approach

The approach follows the general flow depicted in Figure 1.2. The following subsections describe the respective techniques as well as their linkages and dependencies.

Wireless Channel Prediction

Model Architecture

We propose to use a machine learning-based approach that comprises a Long Short-Term Memory (LSTM) Network to solve Problem 2. As shown in Figure 4.1, the input to the LSTM network is a sequence of of the following data recorded at each LSTM time step:

- The current position, orientation, the linear and angular velocity of the i^{th} vehicle $(x_i, y_i, \psi_i, v_i, \dot{\psi}_i)$.
- The current position, orientation and velocity of the leading vehicle $(x_{i-1}, y_{i-1}, \psi_{i_1}, v_{i-1}, \dot{\psi_{i-1}})$.
- the predicted future states of both vehicles over a time horizon T.
- The current measurements of the wireless channel (CSI) collected at the i^{th} vehicle.

The model architecture is a traditional LSTM network that consists of a number of LSTM layers, where each layer consists of LSTM cells connected to each other. The last LSTM layer connects to a fully connected layer with an activation function that generates the final output. The output of the model is the predicted packet delivery rate PDR_i estimated over the time horizon T. We choose to use an LSTM network due

to the promise that has been shown in applying it in several sequence-to-sequence learning tasks [119] like speech recognition and machine translation.

The intuition behind this approach is that the on-board sensors of autonomous vehicles are already being used to capture the static and dynamic features of the physical world, including the dynamic motion related to the vehicles themselves which have a great affect on the dynamics of the wireless channel. Our approach leverages these sensor data, the predicted motion paths of the vehicles, and the current state of the wireless channel H to come up with a prediction of the future packet delivery rate of the wireless channel.

Dataset Generation

In order to generate a dataset for training and validating our model, we use an indoor test-bed as shown in Figure 4.2. The test-bed is a 5mX4m arena covered by a millimeter accurate motion capture system (OptiTrack [1]) and includes racing tracks for small sized robots/miniature vehicles. We use two turtlebots, each equipped with an Intel NUC device that contains a commodity 5300 Wifi adapter used for communication between the two robots. We use the modified firmware [62] of the Wifi adapters in order to capture the Channel State Information (CSI) at the follower turtlebot. The OptiTrack system provides position and orientation information of the two turtlebots. We use the on-board wheel encoders on each turtlebot to provide linear and angular velocity information. All the data is synchronized and collected at a single computing device at the receiver. We develop a data acquisition node under the Robotic Operating System (ROS) framework to collect the data and generate datasets to be used for training and validating the proposed LSTM model. This method of dataset generation can be generalized for platoons of autonomous cars applications by using crowd sourcing techniques, pulling features extracted from each car's sensors and wireless adapters and uploading them to the cloud.

Formal Analysis of Wireless Protocol

Given the predicted PDR in the last section, we use *probabilistic model checking* to convert this probability into packet delivery latency and estimate the best/worst-case performance bounds for the decision making in the motion planning and control. We use *PRISM* [72], a common model checking tool for wireless protocols. We define the wireless communication protocol between vehicles based on CSMA/CA, in which a transmitter tries to avoid collision by sensing the channel. We define p_1 as the probability of channel being idle. In

[1] https://optitrack.com/products/prime-17w/

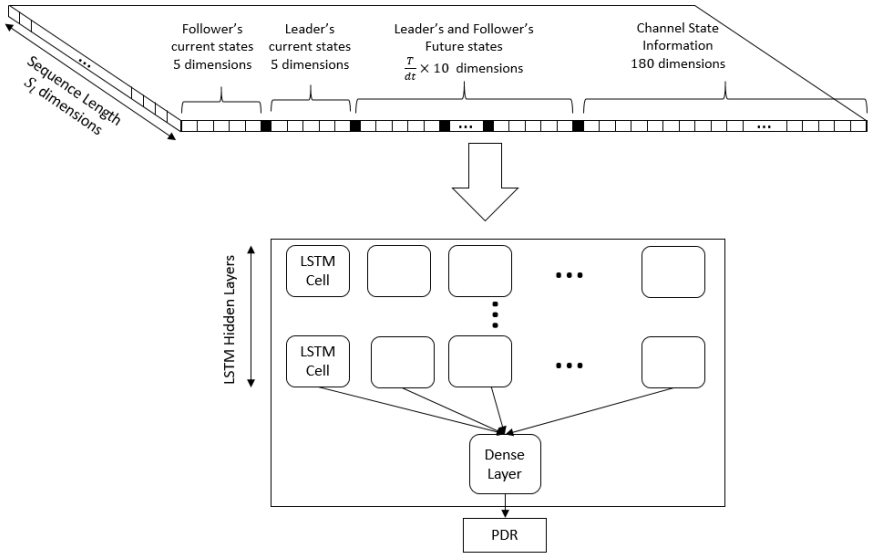


Figure 4.1: LSTM Network Architecture

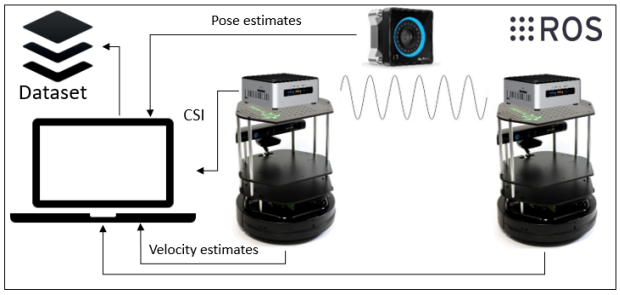


Figure 4.2: Data collection framework for LSTM model training and testing

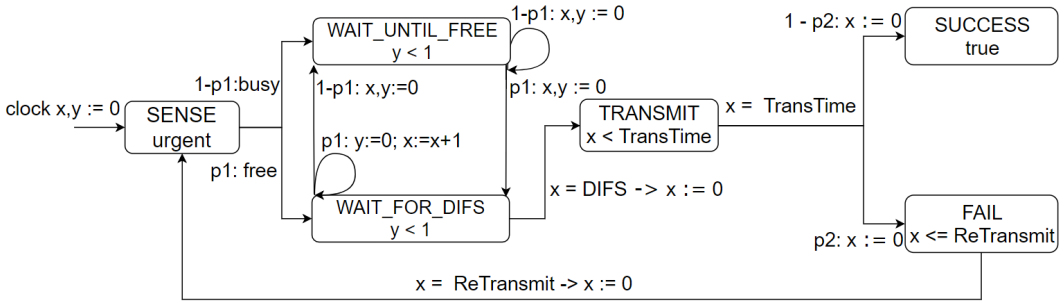


Figure 4.3: Probabilistic model of wireless protocol

addition, in the case of collision or packet loss, the packet will be retransmitted up to the maximum number of attempts, we define p_1 as the packet delivery probability (or PDR).

Figure 4.3 shows the probabilistic model of the wireless communication protocol we defined. The probability of each transition in this model is assumed to be 1 unless indicated specifically. The initial state is indicated

by the arrow. The leader begins with a data packet to send, and senses the channel. With a probability p_1, which is the Clear Channel Assessment (CCA) probability, the channel is free, then the leader would enter into the WAIT_FOR_DIFS state; otherwise, with a probability $1 - p_1$ the channel is busy and the leader would enter into the WAIT_UNTIL_FREE state. If the channel remains free for DIFS = $128us$, then the leader enters its transmission state and starts sending a packet. However, during the waiting process the channel will be free with probability p_1 and busy with probability $1 - p_1$. We assume the time taken to send a packet is $1ms$, which is represented as TransTime in the figure, and the success state of the transmission whether the state SUCCESS or FAIL is entered depends on whether a packet loss has occurred and is recorded by the channel. The probability of delivering a packet successfully is defined as Packet Delivery Rate (PDR) p_2, which means the packet can be sent successfully and the leader vehicle will enter into SUCCESS state with probability p_2. Otherwise, the leader will enter into FAIL sate and the packet is lost in this case. After entering the FAIL state, the leader will wait for some retransmission time, which is represented as ReTransmit in the figure, to send the packet again.

We assume that PDR can be estimated by the wireless channel prediction technique presented above, and CCA can be estimated based on the number of the nearby communicating cars on the road.

The output of this model is the latency-bounded probabilistic reachability which is the probability of leading vehicle correctly delivering its packets within latency bound T.

Motion Planning and Control

Nonlinear Model Predictive Control (NMPC) is an advanced control method that generates an optimal trajectory based on a dynamical model of the system. This technique minimizes the deviation of the predicted trajectory from a given reference signal while keeping all the constraints satisfied. NMPC solves an optimal control problem that is designed for a time horizon with multiple time steps. After finding the optimal trajectory, only the first step of the control strategy is implemented and the optimization process is repeated again, starting from the new state and resulting in a new control strategy and predicted state trajectory. At each time step, the NMPC controller solves the following constrained optimal control problem:

$$\underset{\mathbf{u}_i(.)}{\text{minimize}} \quad \int_{t=t_0}^{t_0+T} ||\mathbf{x}_i(t) - \mathbf{x}_{i-1}(t)||^2_{P_i} \tag{4.8a}$$

$$+ ||\mathbf{x}_i(t) - \mathbf{x}^{ref}(t)||^2_{Q_i} + ||\mathbf{u}_i(t)||^2_{R_i} \tag{4.8b}$$

$$+ ||\mathbf{x}_i(t_0 + T) - \mathbf{x}_{i-1}(t_0 + T)||^2_{S_i} \tag{4.8c}$$

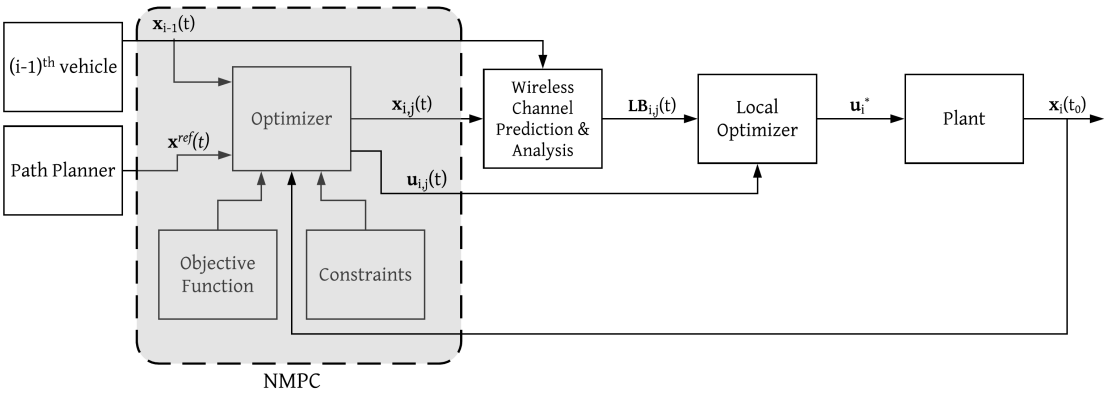


Figure 4.4: Schematic architecture of Controller, $t \in [t_0, t_0 + T]$

$$\text{subject to} \quad \dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t), \mathbf{u}_i(t), t), \forall t \in [t_0, t_0 + T], \tag{4.8d}$$

$$\mathbf{u}_i(t_0) = 0, \tag{4.8e}$$

$$\mathbf{x}_i(t_0) = \mathbf{x}_{i,0}, \tag{4.8f}$$

$$\gamma_i(t) \geqslant \Gamma_{i,j}(t), \forall t \in [t_0, t_0 + T], \tag{4.8g}$$

$$\underline{\mathbf{x}_i} \leqslant \mathbf{x}_i(t) \leqslant \overline{\mathbf{x}_i}, \forall t \in [t_0, t_0 + T], \tag{4.8h}$$

$$\underline{\mathbf{u}_i} \leqslant \mathbf{u}_i(t) \leqslant \overline{\mathbf{u}_i}, \forall t \in [t_0, t_0 + T], \tag{4.8i}$$

where T is the time horizon and t_0 is the current time. Equations (4.8a)-(4.8c) show the objective function, (4.8a) is penalizing the distance of the i^{th} vehicle from its preceding car, and (4.8b) minimizes the amount of fuel consumption and deviation from a reference trajectory $\mathbf{x}^{ref}(.)$. The last equation (4.8c) is the terminal condition. Also, $P_i, Q_i, S_i \succeq 0$, and $R_i \succ 0$ are tuning matrices, or cost coefficients. The first constraint (4.8d) is a dynamical system of the vehicle that is given by equations (4.2)-(4.3). The equations (4.8e) and (4.8f) define the initial states and actions of the agent and the last two constraints (4.8h) and (4.8i) define the bounds for state variables and control inputs. The safety condition is enforced by inequality (4.8g), in which $\Gamma_{i,j}(t)$ is the j^{th} time-to-collision , $\forall j \in \{1, ..., m\}$, which is given as a parameter to the model.

The control algorithm generates different trajectories by changing the value of $\Gamma_{i,j}(t) \in [0.6, 1.6]$ in constraint (4.8g) and then passes $\mathbf{x}_i$ to the Wireless Channel Prediction & Analysis. Subsequently, Wireless Channel Prediction & Analysis predicts the latency bound based on the current state of the $(i-1)^{th}$ vehicle and the received trajectory of i^{th} vehicle from NMPC (figure 4.4). Then a Local Optimizer runs a simple

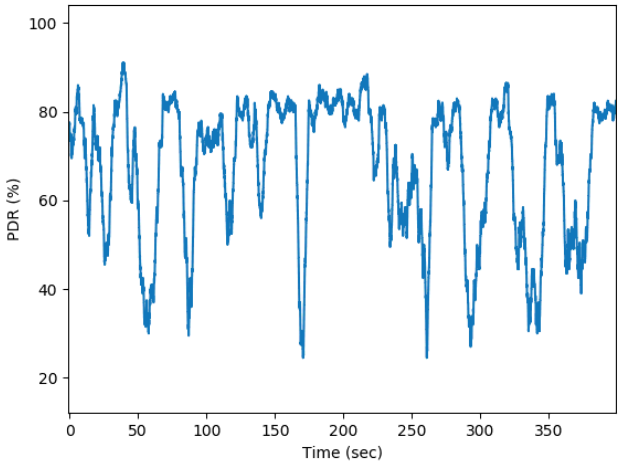


Figure 4.5: Example showing the significant variability in PDR due to a dynamic physical environment.

algorithm to calculate a value for each latency bound according to the following equation

$$\Gamma_i^* = \underset{LB_{i,j}, j=1,\ldots,m}{\operatorname{argmin}} \sum_{t=t_0}^{t_0+T} \alpha^{t-t_0} LB_{t,j}, 0 < \alpha < 1 \tag{4.9}$$

where m is the number of TTC parameters used with associated trajectories and $LB_{t,j}$ denotes the latency bound of predicted trajectory j at time step t. By setting α between zero and one we ensure that the value of latency bound at the first time steps play a more important role than later time steps; that is, α is a discount term on latency predictions. The motivation behind the idea of latency minimization is that larger latencies will negatively impact the performance of the controller.

4.3 Simulations & Experiments

In order to test the performance of the proposed LSTM network in predicting varying PDR values in a dynamic environment, we set up an environment that imposes fluctuations in the wireless channel quality values over time. We achieved that by removing the antennas from the Intel NUC devices and thus reducing the transmission power greatly relative to the area of the testbed arena. Next, we collected data by moving one Turtlebot in random motions around the arena with variable speeds and directions. We used a transmission frequency $f_t = 50Hz$. We recorded the PDR values over a time window $T = 4s$. Figure 4.5 shows that PDR values vary significantly between 20% and 90% as only one Turtlebot moves randomly inside the arena. We used an LSTM network with a sequence length $S_l = 200$, and contains one hidden layer with 64 LSTM cells. We record a dataset of 24,000 data samples. We use 80% of the dataset for training and the rest for validation.

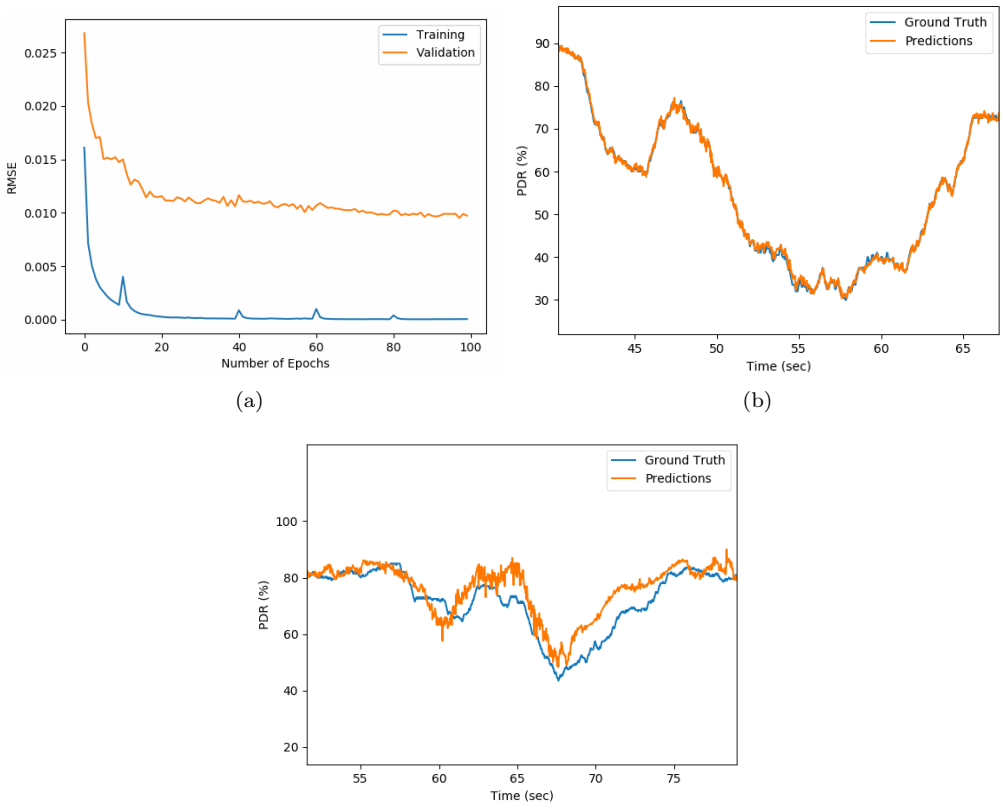


Figure 4.6: (a) The value of the loss function for both training and validation data as the LSTM model is being trained. (b) A zoomed-in snapshot from the training data showing the ability of the trained LSTM model to predict PDR with 98% accuracy. (c) A zoomed-in snapshot from the validation data showing ability of the trained LSTM model to predict PDR with 91% accuracy.

As shown in Figure 4.6, the trained LSTM network achieved a 98% prediction accuracy in training and a 91% prediction accuracy in validation. We compared the performance of our trained LSTM network with a revised version of the KNN-based approach presented in [120] using the same features for prediction. Figure 4.7 shows that our trained LSTM network achieved a median prediction accuracy of $+/-5\%$ which is an order of magnitude better than performance of the kNN-based approach. We repeated the learning process on another dataset that we collected while moving the two turtlebot randomly inside the arena. The trained LSTM network achieved 97% accuracy on the training data and 87% accuracy on the validation data. We believe that this accuracy indicates an over fitting behaviour of the network that may be further enhanced by collecting more data, or performing more tuning of LSTM hyper-parameters (e.g. adding dropout layers, performing regularization).

To examine the control approach presented above, we simulate the changes of the latency bound in a communication channel established between two vehicles while passing beneath a bridge. A bridge causes

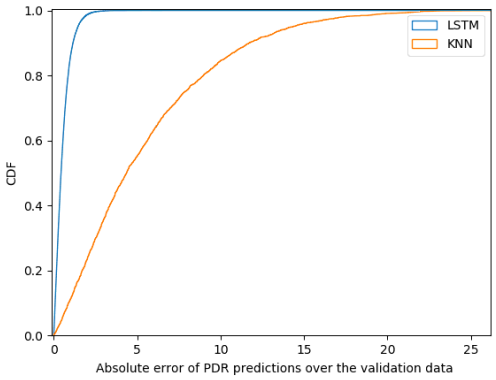
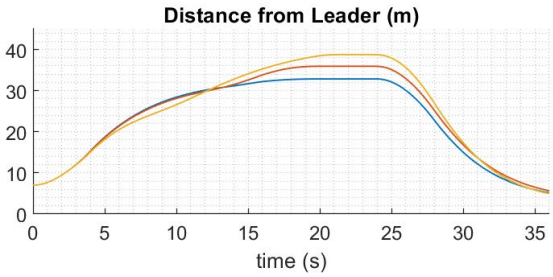
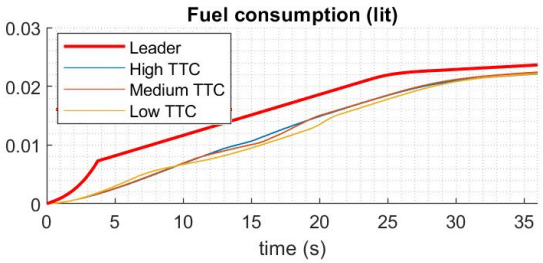


Figure 4.7: The proposed LSTM-based approach achieves a median performance an order of magnitude better than the kNN-based one.

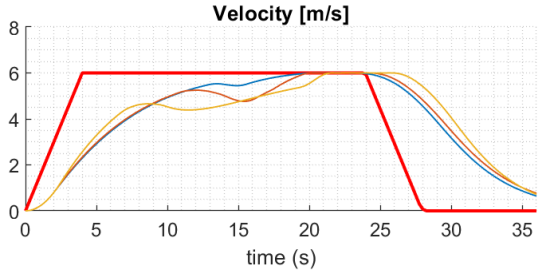
multipath interference, which affects the wireless communication channel and the latency bound will grow, and subsequently impacts the performance of the following vehicle [121]. To simplify the process, we consider three different TTC parameters, $\gamma_i^h(t) = 1.6$,$\gamma_i^m(t) = 1.1$, and $\gamma_i^l(t) = 0.6(s)$ that denote high, medium, and low time-to-collision, respectively. The NMPC algorithm generates three trajectories based on the given time-to-collisions and passes them to the Wireless Channel Prediction & Analysis. This predictor then computes latency bounds for each of these trajectories and feeds them to the Local Optimizer, in which one of $\gamma_i(t) = \{0.6, 1.1, 1.6\}$ and its associated trajectory will be chosen based on their latency bounds. The results for this scenario are in Figure 4.8. According to the results, although selecting lower TTC results in speeding up and driving close to the leading vehicle in the first few steps, later in time, a safety constraint forces the agent to decrease its velocity to guarantee safety. In this case, following higher TTC will result in a better latency bound vector in the given time horizon and keep almost constant distance between the two vehicles.



(a)



(b)



(c)

Figure 4.8: Local Optimizer suggests the trajectory with higher TTC between the different TTCs (legend), which results in better traffic throughput later in the trajectory (a) even though it has a higher time-to-collision threshold, while maintaining the same fuel consumption level (b), and satisfying the enforced constraints on velocity, (c)

Chapter 5

Safety Assurance of NN Controlled Autonomous Systems

5.1 Problem Formulation

We consider the kinematic bicycle model (KBM) as the dynamical model for our autonomous vehicle. However, the usual KBM is defined in terms of the absolute Cartesian position of the vehicle, which is inconsistent with the sensing modalities typically available to an autonomous vehicle. Thus, we instead describe the kinematics in terms of relative position variables that are directly measurable via LiDAR or visual sensors. In particular, the dynamics for the distance to the obstacle, $\|\vec{r}\|$, and the angle of the vehicle with respect to the obstacle, ξ, comprise a system of ordinary differential equations. These quantities describe a state-space model that is given by:

$$\begin{pmatrix} \dot{r} \\ \dot{\xi} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v\cos(\xi-\beta) \\ -\frac{1}{r}v\sin(\xi-\beta) - \frac{v}{\ell_r}\sin(\beta) \\ a \end{pmatrix}; \quad \beta \triangleq \tan^{-1}\left(\frac{\ell_r}{\ell_f+\ell_r}\tan(\delta_f)\right) \tag{5.1}$$

where $r(t) \triangleq \|\vec{r}(t)\|$; a is the linear acceleration input; δ_f is the front-wheel steering angle input[1]; and $\psi + \xi = \tan^{-1}(y/x)$. For the sake of intuition, we note a few special cases: when $\xi = \pm\pi/2$, the vehicle is oriented tangentially to the obstacle, and when $\xi = \pi$ or 0, the vehicle is pointing directly at or away from the obstacle, respectively (see Fig. 5.1). β is an intermediate quantity, an *invertible function* of δ_f.

[1]That is the steering angle can be set instantaneously, and the dynamics of the steering rack can be ignored.

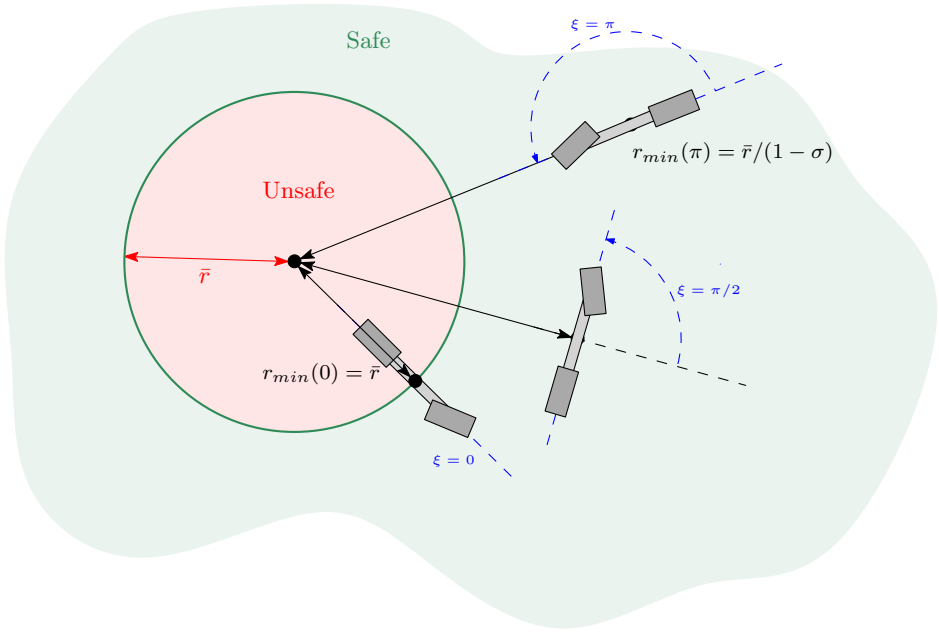


Figure 5.1: Obstacle specification and minimum barrier distance as a function of relative vehicle orientation, ξ.

We make the further assumption that the KBM has a control constraint on δ_f such that $\delta_f \in [-\delta_{f\max}, \delta_{f\max}]$. To simplify further notation, we will consider β directly as a control variable; this is without loss of generality, since there is a bijection between β and the actual steering control angle, δ_f. Thus, β is also constrained: $\beta \in [-\beta_{\max}, \beta_{\max}]$. Finally, we define the state and control vectors for the KBM as: $\chi \triangleq (\xi, r, v)$ and $\omega \triangleq (a, \beta)$, where $\omega \in \Omega_{\text{admis.}} \triangleq \mathbb{R} \times [-\beta_{\max}, \beta_{\max}]$, the set of admissible controls.

Problem 5 *Consider a KBM vehicle with maximum steering angle $\delta_{f\,max}$, length parameters $l_f = l_r$ and maximum velocity v_{max}[2]. Consider also a disk-shaped region of radius $\bar{r}$ centered at the origin, $U = \{x \in \mathbb{R}^2 : \|x\| \leq \bar{r}\}$. Find a set of safe initial conditions, S_0, and a ReLU NN:*

$$\mathscr{NN} : (\chi, \omega) \mapsto \hat{\omega} \tag{5.2}$$

such that for any globally Lipschitz continuous controller $\mu : \chi \mapsto \omega \in \Omega_{admis.}$, the state feedback controller:

$$\mathscr{NN}(\chi, \mu(\chi)) : \chi \mapsto \hat{\omega} \tag{5.3}$$

*is **guaranteed** to prevent the vehicle from entering the unsafe region U if it was started from a state in S_0.*

[2]In our KBM model, this technically requires a feedback controller on a, but this won't affect our results.

Equivalently, applying feedback controller $\mathscr{MN}(\cdot, \mu(\cdot))$ ensures that $r > \bar{r}$ for all time when the initial condition is chosen in S_0.

5.2 Approach

The most important feature of Problem 5 described above is that $\mathscr{MN}$ is a memoryless function that must correct the output of a feedback controller *instantaneously*. The existence of such a corrective function is not a priori guaranteed for the KBM dynamics. However, the well-known theory of Barrier Functions (BFs) provides a mechanism for ensuring the safety of a dynamical systems: in short, barrier functions are real-valued functions of the system state whose properties ensure that the value of the function remains greater than zero along trajectories of the system [122, 123]. Thus, if a barrier function is designed so that its zero super-level set is contained inside the set of safe states, then that subset is forward-invariant; i.e. if the system starts from a safe state, then it will stay safe for all future time. In this way, barrier functions can be used to to convert safety properties into an instantaneous – albeit state-dependent – set membership problem for control actions.

Thus, in the spirit of Problem 5, we employ the usual theory of autonomous barrier functions to control systems under *state-feedback* control: i.e. a control system $\dot{x} = f(x, u)$ in closed loop with a state-feedback controller $\pi : x \mapsto u$. In this scenario, a feedback controller in closed loop converts the control system into an autonomous one – the autonomous vector field $f(\cdot, \pi(\cdot))$. Moreover, the conditions for a barrier function can be translated into a set membership problem for the outputs of such a feedback controller. This is explained in the following corollary.

Corollary 1 *Let $\dot{x} = f(x, u)$ be a control system that is Lipschitz continuous in both of its arguments on a set $\mathcal{D} \times \Omega_{admis.}$; furthermore, let $h : \mathbb{R}^n \to \mathbb{R}$ with $\mathcal{C}_h \triangleq \{x \in \mathbb{R}^n | h(x) \geq 0\} \subseteq \mathcal{D}$, and let α be a class $\mathcal{K}$ function. If the set*

$$R_{h,\alpha}(x) \triangleq \{u \in \Omega_{admis.} | \nabla_x^T h(x) \cdot f(x, u) + \alpha(h(x)) \geq 0\} \tag{5.4}$$

is non-empty for each $x \in \mathcal{D}$, and a feedback controller $\pi : x \mapsto u$ satisfies

$$\pi(x) \in R_{h,\alpha}(x) \quad \forall x \in \mathcal{D} \tag{5.5}$$

then $\mathcal{C}_h$ is forward invariant for the closed-loop dynamics $f(\cdot, \pi(\cdot))$.

Proof: This follows directly from an application of zeroing barrier functions [124, Theorem 1]. □

Corollary 1 is the foundation of ShieldNN: the only difference is that instead of designing a single controller π, we will design a safe "combined" controller $\mathscr{NN}(\cdot, \mu(\cdot))$. In this usage, when a controller μ generates a control action, $\mu(x)$, that lies outside of the set $R(x)$, $\mathscr{NN}$ must map it to a control *within* the set $R(x)$.

Thus, Corollary 1 admits the following three-step framework for developing ShieldNN filters.

ShieldNN Framework:

(1) Design a Candidate Barrier Function. For a function, h, to be a barrier function for a specific safety property, its zero super-level set, $\mathcal{C}_h$, must be contained in the set of safe states.

(2) Verify the Existence of Safe Controls. *(ShieldNN Verifier)* Show that the set $R_{h,\alpha}(x)$ is non-empty for each state $x \in \mathcal{C}_h$. This establishes that a safe feedback controller may exist.

(3) Design a Safety Filter. *(ShieldNN Synthesizer)* If possible, design $\mathscr{NN}_0$ such that $\mathscr{NN}_0 : x \in \mathcal{C}_h \mapsto \hat{u} \in R(x)$; then obtain a safety filter as:

$$\mathscr{NN}(x,u) := \begin{cases} u & \text{if } u \in R(x) \\ \mathscr{NN}_0(x) & \text{if } u \notin R(x). \end{cases} \tag{5.6}$$

ShieldNN thus hinges on the design of a barrier function, and then the design of two **prediction-type** NN functions: $\mathscr{NN}_0$, which generates a **safe** control at each $x \in \mathcal{C}_h$; and $\mathscr{NN}$, which **overrides any unsafe control** for a state with the associated value of $\mathscr{NN}_0$.

5.3 Barrier Function(s) for the KBM Dynamics: the Basis of ShieldNN

It difficult to analytically derive a single barrier function as a function of a particular vehicle and safety radius for the KBM. Thus, we instead define a class of *candidate* barrier functions for a specific vehicle: this class is further parameterized by a unit-less scaling parameter and the safety radius, and it has the property that there are guaranteed parameter choices that actually result in a barrier function. However, since the analytically guaranteed parameter choices are impractically conservative, we devise a ShieldNN verifier algorithm to establish whether a more pragmatic (user-supplied) choice of barrier function parameters does indeed constitute a barrier function.

In particular, we propose the following class of candidate barrier functions to certify control actions so that the vehicle doesn't get within $\bar{r}$ units of the origin (Problem 5):

$$h_{\bar{r},\sigma}(\chi) = h_{\bar{r},\sigma}(\xi, r, v) = \frac{\sigma \cos(\xi/2) + 1 - \sigma}{\bar{r}} - \frac{1}{r} \tag{5.7}$$

where $\sigma \in (0,1)$ is an additional parameter whose function we shall describe subsequently. First note that the equation $h_{\bar{r},\sigma}(\chi) = 0$ has a unique solution, $r_{\min}(\xi)$ for each value of ξ:

$$r_{\min}(\xi) = \bar{r}/(\sigma\cos(\xi/2) + 1 - \sigma), \tag{5.8}$$

so the smallest value of $r_{\min}$ is $r_{\min}(0) = \bar{r}$. Thus, the function $h_{\bar{r},\sigma}$ satisfies the requirements of **(1)** in the ShieldNN framework: i.e. $\mathcal{C}_{h_{\bar{r},\sigma}}$, the zero super-level set of $h_{\bar{r},\sigma}$, is entirely contained in the set of safe states as proscribed by Problem 5, independent of the choice of σ. See Fig. 5.1, which also depicts another crucial value, $r_{\min}(\pm\pi) = \bar{r}/(1-\sigma)$.

Remark 1 *Note that $h_{\bar{r},\sigma}$ is independent of the velocity state, v. This will ultimately force ShieldNN filters to intervene only by altering the steering input.*

A barrier function also requires a class $\mathcal{K}$ function, α. For ShieldNN, we choose a linear function

$$\alpha_{v_{\max}}(x) = K \cdot v_{\max} \cdot x \tag{5.9}$$

where $v_{\max}$ is the assumed maximum linear velocity (see Problem 5), and K is a constant selected according to the following theorem.

Theorem 1 *Consider any fixed parameters $\bar{r}$, ℓ_r and σ. Assume that $0 \leq v \leq v_{max}$ (as specified by Problem 5). If K is chosen such that:*

$$K \geq K_{\bar{r},\sigma} \triangleq \max(\{1, 1/\bar{r}\}) \cdot \left(\tfrac{\sigma}{2\cdot\bar{r}} + 2\right) \tag{5.10}$$

then the Lie derivative $\nabla_\chi^T h_{\bar{r},\sigma}(x) \cdot f_{KBM}(\chi,\omega) + \alpha(h_{\bar{r},\sigma}(\chi))$ is a monotonically increasing function in r for all $r \geq \bar{r}$ for each fixed choice of $v \in (0, v_{max}]$ and the remaining state and control variables.

In particular, for all $\chi \in \mathcal{C}_{h_{\bar{r},\sigma}}$ such that $v \in (0, v_{max}]$ it is the case that:

$$R_{h_{\bar{r},\sigma}}((r_{min}(\xi), \xi, v)) \subseteq R_{h_{\bar{r},\sigma}}(\chi). \tag{5.11}$$

In addition to concretely defining our class of candidate barrier functions, Theorem 1 is the essential facilitator of the ShieldNN algorithm. In particular, note that

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi,\beta,v) \triangleq \left[\nabla_\chi^{\mathrm{T}} h_{\bar{r},\sigma}(\chi) \cdot f_{\mathrm{KBM}}(\chi,\omega) + \alpha(h_{\bar{r},\sigma}(\chi))\right]_{\chi=(r_{\min}(\xi),\xi,v)}$$

$$= v\Big(\tfrac{\sigma}{2\cdot\bar{r}\cdot r_{\min}(\xi)}\sin(\tfrac{\xi}{2})\sin(\xi-\beta) + \tfrac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\tfrac{\xi}{2})\sin(\beta) + \tfrac{1}{r_{\min}(\xi)^2}\cos(\xi-\beta)\Big) \quad (5.12)$$

since $h_{\bar{r},\sigma}((r_{\min}(\xi),\xi,v)) = 0$ and $\alpha_{v_{\max}}(0) = 0$. Hence, the set $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,v))$ is independent of v, so (5.11) gives a sufficient condition for safe controls **(2)** in terms of a single state variable, ξ, and a single control variable β. This simplifies not only the ShieldNN verifier but also the ShieldNN synthesizer, as we shall demonstrate in the next section.

5.4 ShieldNN

5.4.1 ShieldNN Verifier

The overall ShieldNN algorithm has three inputs: the specs for a KBM vehicle ($\ell_f = \ell_r$, $\delta_{f,\max}$ and $v_{\max}$); the desired safety radius ($\bar{r}$); and the barrier parameter σ. From these inputs, the ShieldNN verifier first soundly verifies that these parameters lead to an actual barrier function for Problem 5. As per Theorem 1, it suffices to show that $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot))$ is non-empty for each $\xi \in [-\pi,\pi]$.

If the sets $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot))$ have a complicated structure (both themselves and relative to each other), then establishing this could in principle be quite difficult. However, the barrier functions under consideration actually appear to generate quite nice regions of safe controls. In particular, it appears to the case that the set of safe steering angles in any particular orientation state is an *interval* clipped at the maximum/minimum steering inputs. That is each such set can be written as:

$$R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot)) = \big[\max\{-\beta_{\max}, \mathfrak{l}(\xi)\},\ \min\{\beta_{\max}, \mathfrak{u}(\xi)\}\big], \quad (5.13)$$

where $\mathfrak{l}$ and $\mathfrak{u}$ are continuous functions of ξ. Even more helpfully, the function $\mathfrak{l}$ generally appears to be *concave*, and the symmetry of the problem dictates that $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$. See Fig. 5.2 for an example with parameters $\ell_f = \ell_r = 2$ m, $\bar{r} = 4$ m, $\beta_{\max} = 0.4636$ and $\sigma = 0.48$; $\cup_{\xi\in[-\pi,\pi]} R_{h_{\bar{r},\sigma}}((r_{\min}(\xi),\xi,\cdot))$ is shown in light green, and $\mathfrak{l}$ and $\mathfrak{u}$ are shown in dark green.

Of course these observations about $\mathfrak{l}$ and $\mathfrak{u}$ are difficult to show analytically, given the nature of the equations (c.f. (5.12)). Nevertheless, we can exhibit a sound algorithm to verify these claims for particular parameter values, and hence that the input parameters correspond to a legitimate barrier function as we explain below.

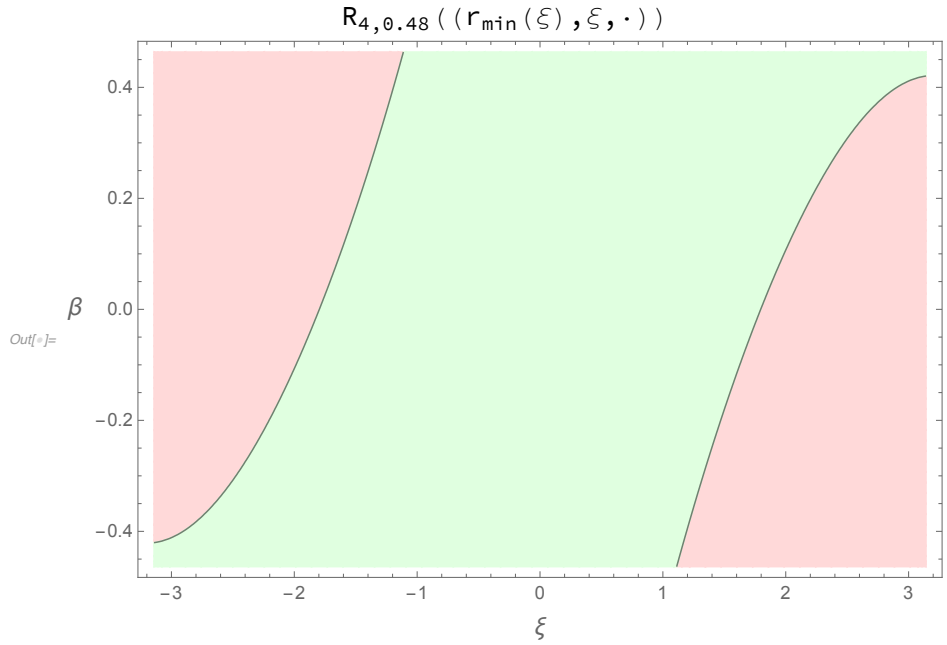


Figure 5.2: Safe/unsafe steering controls. $\cup_\xi R_{4,0.48}((r_{\min}(\xi), \xi, \cdot))$ is shown in light green; $\mathfrak{l}$ and $\mathfrak{u}$ in dark green.

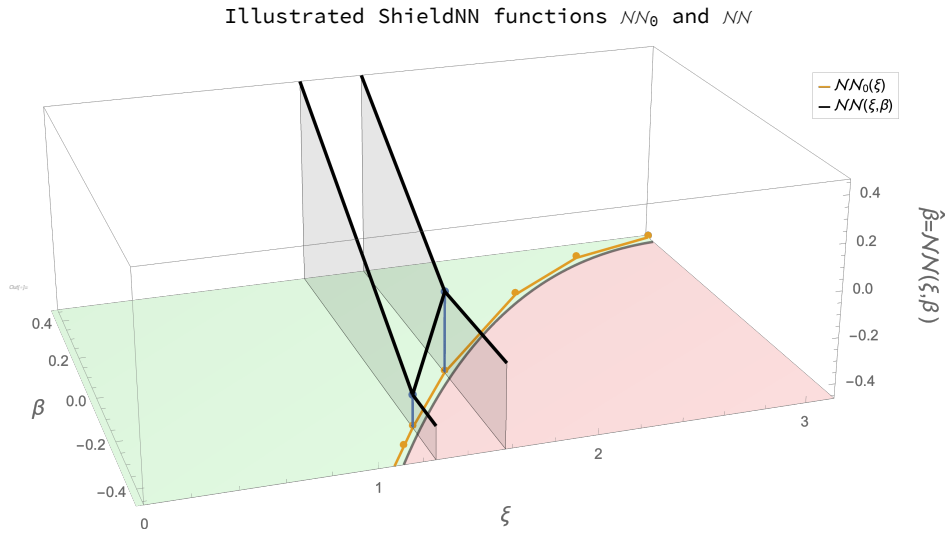


Figure 5.3: Illustration of $\mathscr{NN}_0$ (orange) and two constant-ξ slices of the final ShieldNN filter, $\mathscr{NN}$ (black).

Figure 5.4: Illustrated ShieldNN products for $\ell_f = \ell_r = 2$ m, $\bar{r} = 4$ m, $\beta_{\max} = 0.4636$, $\sigma = 0.48$.

Algorithmic Verification of (5.13):

Recall that the main function of the ShieldNN verifier to soundly verify that equation (5.13) holds for a concave function $\mathfrak{l}$ and with $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$. The conclusion about $\mathfrak{u}$ follows directly from the symmetry of the problem, so we will focus on verifying the claims for $\mathfrak{l}$.

As a foundation for the rest of this subsection, we make the following observation.

Proposition 1 *Suppose that* (5.13) *holds with* $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$. *Then for any* $\xi' \in [-\pi, \pi]$ *such that* $\mathfrak{l}(\xi') \in (-\beta_{max}, \beta_{max})$ *it is the case that*

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi', \mathfrak{l}(\xi'), \cdot) = 0. \tag{5.14}$$

Proof: This follows directly from the definition of $R_{h_{\bar{r},\sigma}}$, and the fact that we are considering it *on* the barrier, i.e. for $\chi' = (r_{\min}(\xi'), \xi', v)$ which implies that $h(\chi') = 0$ and hence that:

$$\alpha_{v_{\max}}(h(\chi')) = 0.$$

□

This suggests that we should start from (5.14) in order to establish the claim in (5.13). To this end, let $a < b$ be real numbers, and define:

$$\mathsf{bd}_{[a,b]} \triangleq \{(\xi', \beta') \in [a, b] \times [-\beta_{\max}, \beta_{\max}] \;\mid\; \mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi', \beta', \cdot) = 0\} \tag{5.15}$$

with the appropriate modifications for other interval types (a, b), $(a, b]$ and $[a, b)$. We also define a related quantity:

$$\mathrm{dom}(\mathsf{bd}_{[a,b]}) = \{\xi \in [a, b] \mid \exists \beta.(\xi, \beta) \in \mathsf{bd}_{[a,b]}\}. \tag{5.16}$$

We can thus develop a sound algorithm to verify (5.13) and the concavity of $\mathfrak{l}$ by soundly verifying the following three properties in sequence:

Property 1. Show that $\mathsf{bd}_{[-\pi,\pi]} \cap \big([-\pi, \pi] \times \{-\beta_{\max}\}\big) = \{(\xi_0, \beta_{\max})\}$; that is $\mathsf{bd}_{[-\pi,\pi]}$ intersects the lower control constraint a single orientation angle, ξ_0. And likewise $\mathsf{bd}_{[-\pi,\pi]} \cap \big([-\pi, \pi] \times \beta_{\max}\big) = \{(-\xi_0, \beta_{\max})\}$ by symmetry.

Property 2. Verify that $\mathsf{bd}_{[\xi_0,\pi]}$ is the graph of a function (likewise for $\mathsf{bd}_{[-\pi,-\xi_0]}$ by symmetry), and that $\mathsf{bd}_{(-\xi_0,\xi_0)} = \emptyset$. Thus, define $\mathfrak{l}$ according to $\mathrm{graph}(\mathfrak{l}) \triangleq \mathsf{bd}_{[\xi_0,\pi]}$.

Property 3. Verify that $\mathfrak{l}$ as defined in **Property 2** is concave.

The ShieldNN verifier algorithm expresses each of these properties as the sound verification that a particular function is greater than zero on a subset of its domain. Naturally, the functions that are associated with these properties are either $\mathcal{L}$ itself or else derived from it (i.e. literally obtained by differentiating), and so each is an analytic function where the variables ξ and β appear only in trigonometric functions. Thus,

these surrogate verification problems are easily approachable by over-approximation and the Mean-Value Theorem.

With this program in mind, the remainder of this appendix consists of one section each explaining how to express **Property 1-3** as minimum-verification problems. These are followed by a section that describes the main algorithmic component of the ShieldNN verifier, `CertifyMin`.

Verifying Property 1

To verify **Property 1**, we can start by using a numerical root finding algorithm to find a zero of $\mathcal{L}(\xi, -\beta_{\max}, \cdot)$, viewed as a function of ξ. However, there is no guarantee that this root, call it $\hat{\xi}_0$ is the only root on the set $[-\pi, \pi] \times \{-\beta_{\max}\}$. Thus, the property to be verified in this case in the assumptions of the following proposition.

Proposition 2 *Suppose that* $\mathcal{L}(\hat{\xi}_0, -\beta_{max}, \cdot) = 0$. *Furthermore, suppose that there exists an* $\epsilon > 0$ *such that:*

(i) $\forall \xi \in [-\pi, \hat{\xi}_0 - \epsilon]$. $\mathcal{L}(\hat{\xi}_0 - \epsilon, -\beta_{max}, \cdot) > 0$;

(ii) $\mathcal{L}(\hat{\xi}_0 - \epsilon, -\beta_{max}, \cdot) > 0$ *and* $\mathcal{L}(\pi, -\beta_{max}, \cdot) < 0$;

(iii) $\forall \xi \in [\hat{\xi}_0 - \epsilon, \pi]$. $\frac{\partial^2}{\partial \xi^2}\mathcal{L}(\xi, -\beta_{max}, \cdot) > 0$.

Then $\hat{\xi}_0$ *is the only root of* $\mathcal{L}(\xi, -\beta_{max}, \cdot)$ *on* $[-\pi, \pi] \times \{-\beta_{max}\}$. *That is* ***Property 1*** *is verified.*

Proof: If *(i)* is true, then there are obviously no zeros of $\mathcal{L}$ on $[-\pi, \hat{\xi}_0 - \epsilon]$.

If *(iii)* is true, then $\mathcal{L}(\xi, -\beta_{\max}, \cdot)$ is a convex function of ξ on the interval $[\hat{\xi}_0, \pi]$. But if *(ii)* is also true, then $\hat{\xi}_0$ must be the only zero of $\mathcal{L}$ on the same interval. This follows by contradiction from the assertion of convexity. If there were another zero on $(\hat{\xi}_0, \pi]$, then the line connecting $(\hat{\xi}_0, \mathcal{L}(\hat{\xi}_0, -\beta_{\max}, \cdot))$ and $(\pi, \mathcal{L}(\pi, -\beta_{\max}, \cdot))$ would lie below this point by assumption *(ii)*, hence contradicting convexity. A similar argument can be made if there were a zero on $[\hat{\xi}_0 - \epsilon, \hat{\xi}_0)$. □

Crucially, the conditions *(i)-(iii)* of Proposition 2 are conditions that can be checked either by verifying that a function is greater than 0 on an interval (as for *(ii)* and *(iii)*), or else that $\mathcal{L}$ as a particular sign for particular inputs (as in *(i)*). Thus, ShieldNN verifier can establish **Property 1** by means of the `CertifyMin` function that we propose later.

Verifying Property 2

Our verification of **Property 2** depends on the conclusion of **Property 1**. In particular, let $\xi_0 = \hat{\xi}_0$ be the single root of $\mathcal{L}$ on $\left([-\pi, \pi] \times \{-\beta_{\max}\}\right)$ as verified above. As before, we provide a proposition that gives us sufficient conditions to assert the conclusion of **Property 2**, and where verifying those conditions requires at worst checking the sign of some $\mathcal{L}$-derived function on an interval (or rectangle).

The main technique for proving that $\mathsf{bd}_{[\xi_0,\pi]}$ is the graph of a function is to note that constant-level curves of $\mathcal{L}$ are solutions to the ODE defined by its gradient. In particular, then, $\mathsf{bd}_{[\xi_0,\pi]}$ contains such a solutions in the rectangle of interest, since it is a subset of the zero-constant level curve of $\mathcal{L}$. Thus, we can verify the desired properties of $\mathsf{bd}_{[\xi_0,\pi]}$ by considering the aforementioned ODE, and demonstrating that it has only one solution in the rectangle of interest. This is the subject of the following proposition and the structure of its subsequent proof.

Proposition 3 *Let ξ_0 be as above. Now suppose the following two conditions are satisfied:*

(i) $\frac{\partial}{\partial \xi}\mathcal{L}(\xi_0, -\beta_{max}, \cdot) > 0$*;*

(ii) for all $(\xi, \beta) \in \left([\xi_0, \pi] \times [-\beta_{max}, \beta_{max}]\right)$ *it is the case that*

$$\frac{\partial}{\partial \beta}\mathcal{L}(\xi, \beta, \cdot) < 0; \tag{5.17}$$

and

(iii) there exists $\epsilon > 0$ *and* $\hat{\beta}_0 \in [-\beta_{max}, \beta_{max}]$ *such that*

(I) $\forall \xi \in [-\beta_{max}, \hat{\beta}_0 - \epsilon]$. $\mathcal{L}(\pi, \hat{\beta}_0 - \epsilon, \cdot) < 0$*;*

(II) $\forall \beta \in [\hat{\beta}_0 - \epsilon, \beta_{max}]$. $\frac{\partial}{\partial \beta}\mathcal{L}(\pi, \beta, \cdot) > 0$.

Then $\mathsf{bd}_{[\xi_0,\pi]}$ *is the graph of a function, and we can define the function* $\mathfrak{l}$ *on* $[\xi_0, \pi]$ *by graph*$(\mathfrak{l}) \triangleq \mathsf{bd}_{[\xi_0,\pi]}$.

Proof: Consider the ODE defined by:

$$\begin{aligned} \dot{\xi} &= -\tfrac{\partial}{\partial \beta}\mathcal{L}(\xi, \beta, \cdot) \\ \dot{\beta} &= \tfrac{\partial}{\partial \xi}\mathcal{L}(\xi, \beta, \cdot). \end{aligned} \tag{5.18}$$

The solutions to (5.18) are guaranteed to exist and be unique on $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$, since the vector field is locally Lipschitz on that rectangle (it is differentiable). Thus, any solution of (5.18) is guaranteed to follow

a constant-level curve of $\mathcal{L}$ within this rectangle; the particular constant-level curve is decided by the value of $\mathcal{L}$ for its initial condition.

As a first step, we establish two facts about the solution of (5.18) with initial condition $(\xi_0, -\beta_{\max})$:

1. the β component of this solution is strictly increasing; and
2. the solution exits $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$ only through its $\xi = \pi$ edge.

First note that some initial portion of this solution must be contained in $[\xi_0, \pi] \times [-\beta_{\max}, \beta_{\max}]$ assumption *(i)* and assumption *(ii)* applied to $(\xi_0, -\beta_{\max})$. Statement 1 is thus established directly by assumption *(ii)*. Now we establish 2. Note that the solution cannot exit via the $\xi = \xi_0$ edge because its β component is strictly increasing. And it can't exit the $\beta = \mp\beta_{\max}$ edges either, because we have verified that $\mathcal{L}$ has only one root on each edge, $(\xi, -\beta_{\max})$ and $(-\xi_0, \beta_{\max})$, respectively. This leaves only the $\xi = \pi$ edge. The solution must leave this rectangle eventually, by the exclusion of the other edges and the fact that its β component is strictly increasing. Thus, it exits via the $\xi = \pi$ edge.

The final conclusion about the functionality follows if $\mathsf{bd}_{[\xi_0,\pi]}$ if $\mathsf{bd}_{[\xi_0,\pi]}$ corresponds *exactly* to the single, unique solution described above. To verify this, we need to verify that there is a single root of $\mathcal{L}$ along the $\xi = \pi$ edge, much as we did to verify **Property 1**; this is made possible by assumption *(iii)*, items *(I)-(II)*. □

As in the case of Proposition 2, the conditions of Proposition 3 are conditions that can be checked using the `CertifyMin` function that we will propose subsequently.

Verifying Property 3

We verify **Property 3** starting from the assumption that verifications of **Property 1** and **Property 2** were successful. In particular, we assume a function $\mathfrak{l}$ with domain $[\xi_0, \pi]$ that defines the lower boundary of the set $R_{h_{\bar{r},\sigma}}$, and which is characterized entirely by $\mathcal{L}(\xi, \mathfrak{l}(\xi), \cdot) = 0$.

Since $\mathfrak{l}$ corresponds exactly to such a constant-level contour, we can use derivatives of $\mathcal{L}$ to compute the derivative of $\mathfrak{l}$ with respect to ξ. That is if we define

$$\gamma'(\xi, \beta) \triangleq -\frac{\partial \mathcal{L}_{\bar{r},\sigma,\ell_r}/\partial \xi}{\partial \mathcal{L}_{\bar{r},\sigma,\ell_r}/\partial \beta}(\xi, \beta) \tag{5.19}$$

then $\mathfrak{l}'(\xi) = \gamma'(\xi, \mathfrak{l}(\xi))$.

By extension then, it is possible to derive the *second* derivative of $\mathfrak{l}$ using $\mathcal{L}$ if we define:

$$\gamma''(\xi, \beta) \triangleq \tfrac{\partial}{\partial \xi}\gamma'(\xi, \beta) + \tfrac{\partial}{\partial \beta}\gamma'(\xi, \beta) \cdot \gamma'(\xi, \beta) \tag{5.20}$$

so that $\mathfrak{l}''(\xi) = \gamma''(\xi, \mathfrak{l}(\xi))$. This gives us an obvious sufficient condition for the *concavity* of $\mathfrak{l}$.

Proposition 4 *Suppose that* ξ_0 *and* $\mathrm{graph}(\mathfrak{l}) \triangleq \mathsf{bd}_{[\xi_0,\pi]}$ *as above. If for all* $(\xi, \beta) \in [\xi_0, \pi] \times [-\beta_{max}, \beta_{max}]$ *we have that*

$$\gamma''(\xi, \beta) < 0 \tag{5.21}$$

then $\mathfrak{l}$ *is concave.*

Proof: Direct from the calculations above. □

5.4.2 Additional Notation

Throughout the rest of this section we will use the following notation:

$$\begin{aligned}\mathscr{L}_{\bar{r},\sigma,\ell_r}(\chi, \beta) &\triangleq \nabla_\chi^{\mathrm{T}} h_{\bar{r},\sigma}(\chi) \cdot f_{\mathrm{KBM}}(\chi, (\beta, a)) \\ &= v\left(\frac{\sigma}{2 \cdot \bar{r} \cdot r}\sin(\xi/2)\sin(\xi - \beta) + \frac{\sigma}{2 \cdot \bar{r} \cdot \ell_r}\sin(\xi/2)\sin(\beta) + \frac{\cos(\xi - \beta)}{r^2}\right).\end{aligned} \tag{5.22}$$

Where f_{KBM} is the right-hand side of the ODE in (5.1) and the variable a is merely a placeholder, since the (5.22) doesn't depend on it at all. In particular, (5.22) has the following relationship with (5.12):

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi, \beta, v) = \mathscr{L}_{\bar{r},\sigma,\ell_r}((r_{\min}(\xi), \xi, v), \beta). \tag{5.23}$$

Moreover, we define the following set:

$$\tilde{\mathcal{C}}_{h_{\bar{r},\sigma}} \triangleq \left\{\chi' = (r', \xi', v') \;\middle|\; h(\chi') \geq 0 \;\wedge\; 0 < v' \leq v_{\max}\right\}, \tag{5.24}$$

which is the subset of the zero-level set of $h_{\bar{r},\sigma}$ that is compatible with our assumption that $0 < v \leq v_{\max}$ (see Problem 5).

5.4.3 Proof of Theorem 1

We prove the first claim of Theorem 1 as the following Lemma.

Lemma 1 *Consider any fixed parameters* $\bar{r}$, ℓ_r, σ *and* $v_{max} > 0$. *Furthermore, define*

$$K_{\bar{r},\sigma} \triangleq \max(\{1, 1/\bar{r}\}) \cdot \left(\tfrac{\sigma}{2\cdot\bar{r}} + 2\right). \tag{5.25}$$

Now suppose that $h_{\bar{r},\sigma}$ is as in (5.7), and $\alpha_{v_{max}}$ is as in (5.9) with K is chosen such that $K \geq K_{\bar{r},\sigma}$.

Then for each $(\xi, v, \beta) \in [-\pi, \pi] \times (0, v_{max}] \times [-\beta_{max}, \beta_{max}]$, the function

$$L_{\xi,v,\beta} : r \in [\bar{r}, \infty) \mapsto \mathscr{L}_{\bar{r},\sigma,\ell_r}((r, \xi, v), \beta) + \alpha_{v_{max}}(h_{\bar{r},\sigma}((r, \xi, v))) \tag{5.26}$$

is increasing on its domain, $\mathrm{dom}(L_{\xi,v,\beta}) = [\bar{r}, +\infty)$.

Remark 2 *Note the relationship between the function $L_{\xi,v,\beta}$ in (5.26) and the function used to to define $R_{h,\alpha}$ in Corollary 1. That is the set that we are interested in characterizing in Theorem 1.*

Proof: We will show that when $K \geq K_{\bar{r},\sigma}$, each such function $L_{\xi,v,\beta}$ has a strictly positive derivative on its domain. In particular, differentiating $L_{\xi,v,\beta}$ gives:

$$\begin{aligned}
\frac{\partial}{\partial r} L_{\xi,v,\beta}(r) &= \frac{\partial}{\partial r}\left[\mathscr{L}_{\bar{r},\sigma,\ell_r}((r, \xi, v), \beta) + \alpha_{v_{\max}}(h_{\bar{r},\sigma}((r, \xi, v)))\right] \\
&= v\left(-\tfrac{\sigma}{2\cdot\bar{r}\cdot r^2}\sin(\xi/2)\sin(\xi-\beta) - 2\tfrac{\cos(\xi-\beta)}{r^3}\right) + \tfrac{K\cdot v_{\max}}{r^2} \\
&\geq v\left(-\tfrac{\sigma}{2\cdot\bar{r}\cdot r^2} - \tfrac{2}{r^3}\right) + \tfrac{K\cdot v_{\max}}{r^2}.
\end{aligned} \tag{5.27}$$

To ensure that this derivative is strictly positive, it suffices to choose K such that

$$v\left(-\frac{\sigma}{2\cdot\bar{r}\cdot r^2} - \frac{2}{r^3}\right) + \frac{K\cdot v_{\max}}{r^2} \geq 0. \tag{5.28}$$

For this, we consider two cases: $\bar{r} < 1$ and $\bar{r} \geq 1$.

When $\bar{r} \geq 1$, then $1/r^3 \leq 1/r^2$ for all $r \geq \bar{r}$. Thus it suffices to choose K such that

$$K \geq \frac{v}{v_{\max}}\left(\frac{\sigma}{2\cdot\bar{r}} + 2\right), \tag{5.29}$$

which is assured under the assumption that $v \in (, v_{\max}]$ if

$$K \geq \frac{\sigma}{2\cdot\bar{r}} + 2. \tag{5.30}$$

Now, when $\bar{r} < 1$, choosing K according to (5.30) ensures that (5.28) is true for all $r \geq 1$. Thus, we also have to ensure (5.28) holds for $\bar{r} \leq r < 1$. But in this case, $1/r^3 \geq 1/r^2$, so (5.28) will be satisfied if

$$K \geq \frac{1}{\bar{r}}\left(\frac{\sigma}{2\cdot\bar{r}} + 2\right). \tag{5.31}$$

Thus, the desired conclusion holds if we choose $K \geq K_{\bar{r},\sigma}$ as defined in the statement of the lemma. □

Now, we have the prerequisites to prove Theorem 1.

Proof: (Theorem 1) The first claim of Theorem 1 is proved as Lemma 1. Thus, it remains to show that for any $\chi = (r, \xi, v) \in \mathcal{C}_{h_{\bar{r},\sigma}}$ with $v \in (0, v_{\max}]$ — that is $\chi \in \tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$ — we have that (5.11) holds. However, this follows from Lemma 1.

In particular, choose an arbitrary $\chi' = (r', \xi', v') \in \tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$, and choose an arbitrary $\omega' = (\beta', a') \in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'), \xi', v'))$; as usual we will only need to concern ourselves with the steering control, β'. First, observe that by definition:

$$(\beta', a') \in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'), \xi', v')) \\ \implies \mathscr{L}_{\bar{r},\sigma,\ell_r}((r_{\min}(\xi'), \xi', v'), \beta') + \alpha_{v_{\max}}(h_{\bar{r},\sigma}((r_{\min}(\xi'), \xi', v'))) \geq 0. \quad (5.32)$$

However, the conclusion of this implication can be rewritten using the definition (5.26):

$$(\beta', a') \in R_{h_{\bar{r},\sigma}}((r_{\min}(\xi'), \xi', v')) \implies L_{\xi',v',\beta'}(r_{\min}(\xi')) \geq 0. \quad (5.33)$$

We now invoke Lemma 1: since $r_{\min}(\xi') \geq \bar{r}$ by construction, Lemma 1 indicates that $L_{\xi',v',\beta'}$ is strictly increasing on the interval $[r_{\min}(\xi'), r']$. Combining this conclusion with (5.33), we see that $L_{\xi',v',\beta'}(r') \geq 0$. Again using the definition of $L_{\xi',v',\beta'}$ in (5.33), we conclude that

$$\mathscr{L}_{\bar{r},\sigma,\ell_r}((r', \xi', v'), \beta') + \alpha_{v_{\max}}(h_{\bar{r},\sigma}(r', \xi', v'))) \geq 0. \quad (5.34)$$

Thus, we conclude that $(\beta', a') \in R_{h_{\bar{r},\sigma}}(\chi')$ by the definition thereof (see the statement of Theorem 1). Finally, since χ' and ω' were chosen arbitrarily, we get the desired conclusion. □

5.4.4 Proof of That a Barrier Function Exists for Each KBM Instance

For $h_{\bar{r},\sigma}$ and $\alpha_{v_{\max}}$ to be a useful class of barrier functions, it should be that case that at least one of these candidates is in fact a barrier function for each instance of the KBM. We make this claim in the form of the following Theorem.

Theorem 2 *Consider any KBM robot with length parameters $\ell_r = \ell_f$; maximum steering angle $\delta_{f_{max}}$; and maximum velocity $v_{max} > 0$. Furthermore, suppose that the following two conditions hold:*

(i) $\beta_{max} \leq \pi/2$, *or equivalently,* $\delta_{f\,max} \leq \frac{\pi}{2}$;

(ii) $\frac{1}{\ell_r}\left(\sigma(1-\sigma)\ell_r + \sigma\bar{r}\right)\sin(\frac{\pi}{4} + \frac{\beta_{max}}{2})\sin(\beta_{max}) \geq 2$; *and*

Then for every $\chi = (r, \xi, v)$ *such that* $0 < v \leq v_{max}$ *the set* $R_{h_{\bar{r},\sigma}}(\chi)$ *is non-empty. In particular, the feedback controller (interpreted as a function of* ξ *only):*

$$\pi : \xi \mapsto \begin{cases} -\beta_{max} & \xi < -\epsilon \\ \xi & \xi \in [-\beta_{max}, \beta_{max}] \\ \beta_{max} & \xi > \epsilon \end{cases} \tag{5.35}$$

is safe.

Remark 3 *Note that there is always a choice of* $\bar{r}$ *and* $\sigma \in (0,1)$ *such that condition (ii) can be satisfied. In particular, it suffices for* $\bar{r}$ *and* σ *to be chosen such that:*

$$2\frac{(\ell_r/\bar{r})}{\sin(\beta_{max})\sin(\pi/4 + \beta_{max}/2)} \leq \sigma. \tag{5.36}$$

Thus, by making $\bar{r}$ *large enough relative* ℓ_r*, it is possible to choose a* $\sigma \in (0,1)$ *such that the inequality* (5.36) *holds, and (ii) is satisfied.*

Proof: *(Theorem 2)* As a consequence of Theorem 1, it is enough to show that $R_{h_{\bar{r},\sigma}}((r_{\min}(\xi), \xi, v_{\max}))$ is non-empty for every $\xi \in [-\pi, \pi]$.

The strategy of the proof will be to consider the control $\beta = \pi(\xi)$, and verify that for each $\chi = (r, \xi, v) \in \tilde{\mathcal{C}}_{h_{\bar{r},\sigma}}$ such that $\xi \in [0, \pi]$, we have:

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi, \pi(\xi)) \geq 0. \tag{5.37}$$

The symmetry of the problem will allow us to make a similar conclusion for $\xi \in [-\pi, 0]$.

We proceed by partitioning the interval $[0, \pi]$ into the following three intervals:

$$I_1 \triangleq [0, \beta_{\max}], \qquad I_2 \triangleq (\beta_{\max}, \pi/2 + \beta_{\max}], \qquad I_3 \triangleq (\pi/2 + \beta_{\max}, \pi].$$

and consider the cases that ξ is in each such interval separately.

Case 1 $\xi \in I_1$**:** In this case, $\pi(\xi) = \xi$, and $\xi \leq \beta_{\max} \leq \pi/2$ by assumption. It is direct to show that:

$$\cos(\xi - \pi(\xi)) = \cos(0) \geq 0 \tag{5.38}$$

and

$$\sin(\xi/2)\sin(\xi - \pi(\xi)) = 0. \tag{5.39}$$

Hence, the cos term in (5.12) can be lower bounded by zero, and the first term in (5.12) is identically zero by (5.39). Thus, in this case, (5.12) is lower bounded as as:

$$\mathcal{L}_{\bar{r},\sigma,\ell_r}(\xi, \beta_{\max}) \geq \frac{\sigma \cdot v \cdot \sin(\xi/2)\sin(\pi(\xi))}{2 \cdot \bar{r} \cdot \ell_r}, \tag{5.40}$$

which of course will be greater than zero since $\xi \in I_1 = [0, \beta_{\max}]$ with $\beta_{\max} \leq \pi/2$ by assumption *(i)*.

Case 2 $\xi \in I_2$**:** In this case, $\pi(\xi) = \beta_{\max}$. Thus, for $\xi \in I_2$, we have that:

$$\cos(\xi - \beta_{\max}) \geq 0 \tag{5.41}$$

$$\sin(\xi/2)\sin(\xi - \beta_{\max}) \geq 0 \tag{5.42}$$

$$\sin(\xi/2)\sin(\beta_{\max}) \geq 0. \tag{5.43}$$

Consequently, (5.37) is automatically satisfied, since all of the quantities in the Lie derivative are positive.

Case 3 $\xi \in I_3$**:** In this case, $\pi(\xi) = \beta_{\max}$ as in Case 2. However, the cos term is now negative in this case:

$$0 > \cos(\xi - \beta_{\max}) \geq -\frac{1}{\bar{r}^2}. \tag{5.44}$$

Thus, since the other two terms are positive on this interval, we need to have:

$$\sin(\tfrac{1}{2}(\tfrac{\pi}{2} + \beta_{\max}))\left(\tfrac{\sigma(1-\sigma)}{2\cdot\bar{r}^2}\sin(\pi - \beta) + \tfrac{\sigma}{2\cdot\bar{r}\cdot\ell_r}\sin(\beta)\right) \geq \tfrac{1}{\bar{r}^2}. \tag{5.45}$$

This follows because on I_3, $\sin(\frac{\xi}{2}) \geq \sin(\frac{1}{2}(\frac{\pi}{2} + \beta_{\max}))$ and $\sin(\xi - \beta_{\max}) \geq \sin(\pi - \beta_{\max})$; i.e. we substituted the lower and upper end points of I_3, respectively. Noting that $\sin(\pi - \beta_{\max}) = \sin(\beta_{\max})$, we finally obtain:

$$\sin(\tfrac{1}{2}(\tfrac{\pi}{2} + \beta_{\max}))\sin(\beta_{\max})\left(\tfrac{\sigma(1-\sigma)}{2} + \tfrac{\sigma\bar{r}}{2\cdot\ell_r}\right) \geq 1. \tag{5.46}$$

The preceding is just another form of *(ii)* so we have the desired conclusion in (5.37).

The conclusion of the theorem then follows from the combined consideration of Cases 1-3 and Theorem 1 as claimed above. □

`CertifyMin` and the ShieldNN Verifier

We have listed a number of conditions, which when verified together, are sufficient to prove that a particular set of parameters leads $h_{\bar{r},\sigma}$ to be a barrier function for the KBM. Furthermore, each of these conditions involves asserting that $\mathcal{L}$ or its derivatives are strictly positive or negative on an interval or a rectangle.

Since $\mathcal{L}$ is composed of relatively simple functions, it is possible for a computer algebra system (CAS) to not only obtain each of these verification functions automatically, but to further differentiate each of them once *more*. Thus, we can combine an extra derivative with the Mean-Value Theorem to verify each of these individual claims. We describe this procedure as `CertifyMin` below.

5.4.5 ShieldNN Synthesizer

Given a verified barrier function, recall that synthesizing a ShieldNN filter requires two components: $\mathscr{NN}_0$ and $\mathscr{NN}$. That is $\mathscr{NN}_0$ chooses a *safe* control for each state, and $\mathscr{NN}$ overrides any *unsafe* controls with the output of $\mathscr{NN}_0$.

Design of $\mathscr{NN}_0$. This task is much easier than it otherwise would be, since the ShieldNN verifier also verifies the safe controls as lying between the continuous functions $\max\{-\beta_{\max}, \mathfrak{l}\}$ and $\min\{\beta_{\max}, \mathfrak{u}\}$ where $\mathfrak{l}$ and is concave and $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$. In particular, then, it is enough to design $\mathscr{NN}_0$ as any neural network such that

$$\max\{-\beta_{\max}, \mathfrak{l}\} \leq \mathscr{NN}_0 \leq \min\{\beta_{\max}, \mathfrak{u}\}. \tag{5.47}$$

This property can be achieved in several ways, including training against samples of $\max\{-\beta_{\max}, \mathfrak{l}\}$ for example. However, we chose to synthesize $\mathscr{NN}_0$ directly in terms of tangent line segments to $\mathfrak{l}$ (and thus exploit the *concavity* of $\mathfrak{l}$). A portion of just such a function $\mathscr{NN}_0$ is illustrated by the orange line in Fig. 5.3.

Algorithm 2: `CertifyMin`.

input : function $f(\xi, \beta)$ that is either $\mathcal{L}$ or one of its derivatives; ξ-interval $[\xi_\ell, \xi_h]$; β-interval $[\beta_\ell, \beta_h]$; a sign $s = \pm 1$. **Either $\xi_\ell = \xi_h$ or $\beta_\ell = \beta_h$ may be true but not both.**

output : N_est

```
1  function CertifyMin(f, ξ_ℓ,ξ_h, β_ℓ, β_h, s)
2      Df ← SymbolicGradient(f)
3      if ξ_ℓ = ξ_h then
4          DfNorm ← SymbolicNorm(SymbolicGetComponent(Df, β))
5      else if β_ℓ = β_h then
6          DfNorm ← SymbolicNorm(SymbolicGetComponent(Df, ξ))
7      else
8          DfNorm ← SymbolicNorm(Df)
9      end
10     gridSize ← 1
11     refine ← True
12     while refine do
13         gridSize ← gridSize/10
14         refine ← False
15         for (ξ', β') in GridIterator(gridSize,ξ_ℓ,ξ_h, β_ℓ, β_h) do
16             if s · f(ξ', β') < 0 then
17                 return False
18             else if s · f(ξ', β') < √2· gridSize · DfNorm(ξ', β') then
19                 refine ← True
20                 break
21             end
22         end
23     end
24     return True
25 end
```

Design of $\mathscr{NN}$. Since the value of $\mathscr{NN}_0$ is designed to lie inside the interval of safe controls, the function $\mathscr{NN}_0$ can itself be used to decide when an unsafe control is supplied. In particular, using this property and the symmetry $\mathfrak{u}(\xi) = -\mathfrak{l}(-\xi)$, we can simply choose

$$\mathscr{NN} : \beta \mapsto \min\{\max\{\mathscr{NN}_0(\beta), \beta\}, -\mathscr{NN}_0(-\beta)\}. \tag{5.48}$$

Note: in this construction, the closer $\mathscr{NN}_0$ approximates its lower bound, $\max\{-\beta_{\max}, \mathfrak{l}\}$, the less intrusive the safety filter will be. Two constant-ξ slices of such a $\mathscr{NN}$ are shown in Fig. 5.3.

5.4.6 Extending ShieldNN to Multiple Obstacles

In this section, we consider the general case of multiple obstacles in the environment. We propose two approaches: 1) a Single-Obstacle Safe-Action (SOSA). 2) a Multi-Obstacle-Safe-Action (MOSA) approach which is described in Algorithm 3.

SOSA

We start by sorting the detected obstacle states during runtime according to their distances r from the vehicle. Then, we choose the closest obstacle and directly apply the synthesized ShieldNN $\mathscr{NN}$ to the controller steering action in order to generate safe steering control actions that are safe considering only the current closest obstacle to the vehicle.

MOSA

The goal is to search inside the state-action space for an action that is safe for all detected obstacles in the environment. Let $\Xi = \{\xi_1, \xi_2, ...\xi_N\}$ is the list of angles ξ for all detected obstacles sorted by the closest to the furthest obstacle to the vehicle. First, we get the lower and upper bounds of the safe action interval for each ξ_i as follows: i) if $\xi_i > 0$, then the corresponding safe action interval is $[-\beta_{\max}, \mathscr{NN}_0(\xi)]$ otherwise, the safe action interval is $[-\beta_{\max}, -\mathscr{NN}_0(-\xi)]$. This is due to the symmetry of the boundary between safe and unsafe regions as shown in Fig. 5.2. Second, we find the common safe action interval between all detected obstacles by getting the intersection of these intervals. If the intersection is an empty interval, this means that we cannot find an action that is guaranteed to be safe for all detected obstacles. In that case, we fall back to SOSA, generate a safe action considering the closest obstacle $\mathscr{NN}(\boldsymbol{\Xi}[0])$ and raise a flag indicating that no safe action can be found. If the unsafe controller is a learning based controller (e.g. a neural network), the unsafe action flag is used to train the unsafe controller on avoiding moving the vehicle to a state where

no common safe action for all detected obstacles can be found. In case we find an intersection interval, we generate a safe control action β^* inside the intersection interval that is closest to the input unsafe action β.

Algorithm 3: `MOSA`.

input : Array $\boldsymbol{\Xi}$ contains the list of ξ angles for all detected obstacles sorted by the closest to furthest from the vehicle, β, $\beta_{\max}$, $\mathscr{NN}_0, \mathscr{NN}$

output : β^*, SafeAction

```
1  function MOSA(Ξ, β, β_max, NN_0, NN)
2      SafeIntervalsLBs, SafeIntervalsUBs ← new Array
3      foreach ξ ∈ Ξ do
4          if ξ > 0 then
5              SafeIntervalsLBs.insert(NN_0(ξ))
6              SafeIntervalsUBs.insert(β_max)
7          else
8              SafeIntervalsLBs.insert(−β_max)
9              SafeIntervalsUBs.insert(NN_0(ξ))
10         end
11         IntersectionLB ← max(SafeIntervalsLBs)
12         IntersectionUB ← min(SafeIntervalsUBs)
13     end
14     if IntersectionUB ≥ IntersectionLB then
15         SafeAction ← true
16         if β ∈ [IntersectionLB, IntersectionUB] then
17             β* ← β
18         else if β > IntersectionUB then
19             β* ← IntersectionUB
20         else
21             β* ← IntersectionLB
22         end
23     else
24         β* ← NN(Ξ[0], β), SafeAction ← false
25     end
26     return β*, SafeAction
27 end
```

5.5 ShieldNN Evaluation

We conduct a series of experiments to evaluate ShieldNN's performance when applied to unsafe RL controllers. The CARLA Simulator [125] is used as our RL environment, and we consider an RL agent whose goal is to drive a simulated vehicle while avoiding the obstacles in the environment. The goals of the experiments are to assess the following:

1. The effect of ShieldNN when applied during RL training (Experiment 1) in terms of the average collected reward, obstacle avoidance, etc.

2. The safety of the RL agent when ShieldNN is applied after training (Experiment 2).

3. The robustness of ShieldNN when applied in a different environment than that used in training (Experiment 3).

4. The effect of applying SOSA and MOSA aproaches on the RL agent in case of having an environment with multiple obstacles in terms of safety (Experiment 4).

RL Task: The RL task is to drive a simulated four-wheeled vehicle from point A to point B on a curved road that is populated with obstacles. The obstacles are static CARLA objects randomly spawned at different locations between the two points. We define unsafe states as those in which the vehicle hits an obstacle. As ShieldNN is designed for obstacle avoidance, we do not consider the states when the vehicle hits the sides of the roads to be unsafe with respect to ShieldNN. Technical details and graphical representations are included in the Supplementary Materials.

Reward function and termination criteria: If the vehicle reaches point B, the episode terminates, and the RL agent gets a reward value of a 100. The episode terminates, and the agent gets penalized by a value of a 100 in the following cases: when the vehicle (i) hits an obstacle; (ii) hits one of the sides of the road; (iii) has a speed lower than 1 KPH after 5 seconds from the beginning of the episode; or (iv) has a speed that exceed the maximum speed (45 KPH). The reward function is a weighted sum of four terms, and the weights were tuned during training. The four terms are designed in order to incentivize the agent to keep the vehicle's speed between a minimum speed (35 KPH) and a target speed (40 KPH), maintain the desired trajectory, align the vehicle's heading with the direction of travel, and keep the vehicle away from obstacles. Formally, the reward function is defined as:

$$R(s) = W_s * R_{Speed} + W_c * R_{Centering} + W_a * R_{Angle} + W_d * R_{DistToObst},$$

where:

$$R_{Speed} = \begin{cases} \frac{v}{v_{min}}, & v < v_{min} \\ 1 - \frac{v_{target}}{v_{max}}, & v > v_{target} \\ 1 & otherwise \end{cases}$$

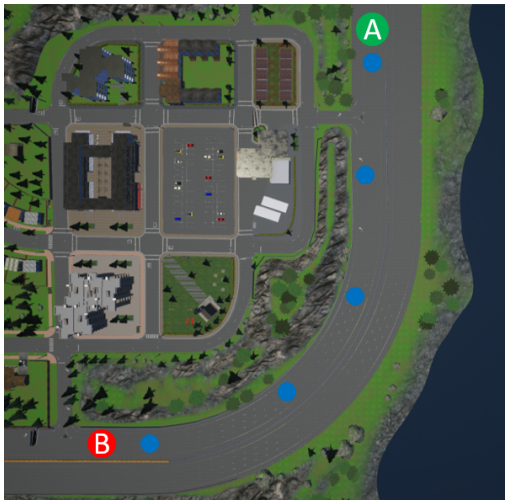


Figure 5.5: RL Task: Goal is to drive the vehicle from point A to point B without hitting random obstacles (the blue circles) spawned along the route

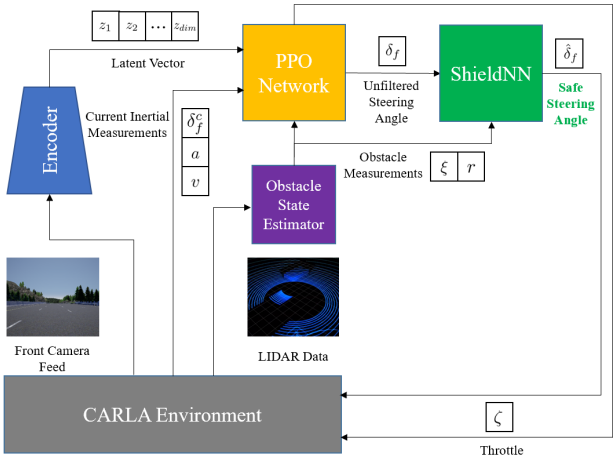


Figure 5.6: Integration Framework of ShieldNN with PPO inside a CARLA simulator Environment.

Figure 5.7: Environment Setup and Integration Framework

$$R_{Centering} = \max\left(1 - \frac{d_c}{l_{max}}, 0\right)$$

$$R_{Angle} = \max\left(1 - \left|\frac{a_c}{\frac{\pi}{9}}\right|, 0\right)$$

$$R_{DistToObst} = \max\left(\min(\frac{r}{r_{max}}, 1), 0\right)$$

v is the current vehicle's speed, d_c is the current lateral distance between the center of the vehicle and the track, a_c is the current angle between the heading of the vehicle and the tangent of the track's curvature,

r is the distance between the center of the vehicle and the nearest obstacle, $r = 0$ if obstacle doesn't exist, $l_{max} = 10m, r_{max} = 20m, v_{min} = 35KPH, v_{max} = 45KPH, v_{target} = 40KPH$.

Integrating ShieldNN with PPO: We train a Proximal Policy Optimization (PPO) [126] neural network in order to perform the desired RL task. To speed up policy learning as in [127], we encode the front camera feed into a latent vector using the encoder part of a trained β-Variational Auto-Encoder (β-VAE). As shown in Fig. 5.6, the encoder takes 160x80 RGB images generated by the simulated vehicle's front facing camera and outputs a latent vector that encodes the state of the surroundings. The inputs to the PPO Network are: The latent vector $[z_1, ..., z_{dim}]$, the vehicle's inertial measurements (current steering angle δ_f^c, speed v and acceleration a) and the relative angle ξ and distance r between the vehicle and the nearest obstacle. The latter two measurements are estimated using an obstacle detection module that takes the vehicle's LIDAR data as input. In our experiments, we assume we have a perfect obstacle detection estimator and we implement it by collecting the ground truth position and orientation measurements of the vehicle and the obstacles from CARLA then calculating ξ and r. The PPO network outputs the new control actions: Throttle ζ and steering angle δ_f. We omit using the brakes as part of the control input vector, as it is not necessary for this task. However, the RL agent will still be able to slow down the vehicle by setting the throttle value to 0 due to the simulated wheel friction force in CARLA. The throttle control action ζ gets passed directly to CARLA, while the steering angle control action gets filtered by ShieldNN. The filter also takes ξ and r as input and generates a new safe steering angle $\hat{\delta_f}$. To train the VAE, we first collect 10,000 images by driving the vehicle manually in CARLA along the desired route with obstacles spawned at random locations and observing different scenes from different orientations. We train the VAE encoder with cross validation and early-stopping. Then, after convergence, we visually inspect the reconstructed image to test the accuracy of the VAE encoder.

ShieldNN Parameters: The ShieldNN filter is synthesized according to Section 5.4 with the parameters $\bar{r} = 4$ m, $\sigma = 0.48$ and KBM parameters $\delta_{f_{\max}} = \pi/4, l_f = l_r = 2m$, and $v_{\max} = 20m/s$.

5.5.1 Experiment 1: Effect of ShieldNN During RL Training

The goal of this experiment is to study the effect of applying ShieldNN to an RL agent during training. We train three RL agents for 6000 episodes each in order to compare (i) the collected reward and (ii) the obstacle hit rate after an equal number of training episodes. The three agents are characterized as follows: Agent 1 is trained with no obstacles and without the ShieldNN filter in place (Obstacles OFF + Filter OFF); Agent 2 is trained with obstacles spawned at random but without ShieldNN in place (Obstacles ON + Filter OFF); and

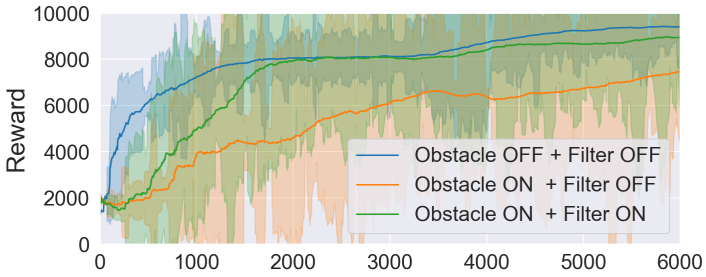


Figure 5.8: Reward (raw & smoothed data for 3 cases)

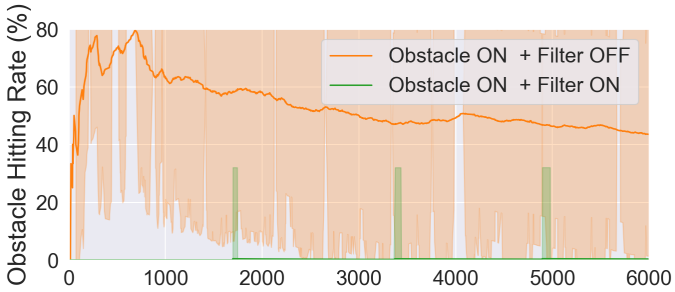


Figure 5.9: Obstacle collision rate

Figure 5.10: Results of Experiment 1, evaluation of effect of ShieldNN during training.

Agent 3 is trained with obstacles spawned at random and with the ShieldNN filter in place (Obstacles ON + Filter ON).

When obstacles are not present (Agent 1), the RL agent quickly learns how to drive the vehicle, as indicated by the rapid growth in the reward function shown in Fig. 5.8. When obstacles are present but ShieldNN is not used (Agent 2), the RL agent's ability to learn the task degrades, as indicated by a 30% reduction in collected reward. However, when obstacles are present and the ShieldNN filter is in place (Agent 3), the agent collects 28% more reward on average than Agent 2, and collects a similar amount of reward to Agent 1. This is an indication that ShieldNN filters improves the training of the system by reducing the number of episodes that are terminated early due to collisions.

Similar behavior can be observed in Fig. 5.9, which shows the obstacle collision rate (averaged across episodes). This figure shows that Agent 2 is slowly learning how to avoid obstacles, since its average obstacle collision rate decreases from 80% to 47% in 60000 episodes. However, Agent 3, which uses ShieldNN during training, has an obstacle collision rate of almost zero. In total, Agent 3 suffers only three collisions across all 60000 episodes. We believe that these three collisions are due to the discrepancy between the KBM and the dynamics of the vehicle used by the CARLA simulator.

	Training		Testing	Experiment 2		Experiment 3A	
Config	**Obstacle**	**Filter**	**Filter**	**TC%[1]**	**OHR%[2]**	**TC%[1]**	**OHR%[2]**
1	OFF	OFF	OFF	7.59	99.5	27.53	79.5
2	OFF	OFF	ON	98.82	0.5	98.73	0.5
3	ON	OFF	OFF	94.82	8.5	71.88	34
4	ON	OFF	ON	100	0	100	0
5	ON	ON	OFF	62.43	44	50.03	60
6	ON	ON	ON	100	0	100	0

[1] TC% := Track Completion % [2] OHR% := Obstacle Hit Rate %

Table 5.1: Experiment 2 & 3, evaluation of safety and performance with and without ShieldNN.

5.5.2 Experiment 2: Safety Evaluation of ShieldNN

The goal of this experiment is to validate the safety guarantees provided by ShieldNN when applied to non-safe controllers. To do this, we evaluate the three trained agents from Experiment 1 in the same environment they were trained in, and with obstacles spawned randomly according to the same distribution used during training. With this setup, we consider two evaluation scenarios: (i) when the ShieldNN filter is in place (ShieldNN ON) and (ii) when ShieldNN filter is not in place (ShieldNN OFF). Table 5.1 shows all six configurations of this experiment. For each configuration, we run 200 episodes and record three metrics: (i) the minimum distance between the center of the vehicle and the obstacles, (ii) the average percentage of track completion, and (iii) the percentage of hitting obstacles across the 200 episodes.

Fig. 5.11 and 5.12 show the histograms of the minimum distance to obstacles for each configuration. The figure also show two vertical lines at 2.3 m and 4 m: the former is the minimum distance at which a collision can occur, given the length of the vehicle, and the latter is the value of the safe distance $\bar{r}$ used to design the ShieldNN filter. Whenever the ShieldNN was not used in the 200 testing episodes (ShieldNN OFF, Fig. 5.11), the average of all the histograms is close to the 2.3 m line indicating numerous obstacle collisions. The exact percentage of obstacle hit rate is reported in Table Table 5.1. Upon comparing the histograms in Fig. 5.11 with those in 5.12, we conclude that ShieldNN nevertheless renders all the three agents safe: note that the center of mass of the histograms shifts above the safety radius parameter, $\bar{r}$, used to design the ShieldNN filter. In particular, Agents 2 and 3 were able to avoid all the obstacles spawned in all 200 episodes, while Agent 1 hit only 0.5% of the obstacles spawned. Again, we believe this is due to the difference between the KBM used to design the filter and the actual dynamics of the vehicle. In general, the obstacle hitting rate is reduced by $99.4\%, 100\%$ and 100% for Agents 1, 2, and 3, respectively.

5.5.3 Experiment 3: Robustness of ShieldNN in Different Environments

The goal of this experiment is to test the robustness of ShieldNN when applied inside a different environment than the one used to train the RL agents. We split the experiment into two parts:

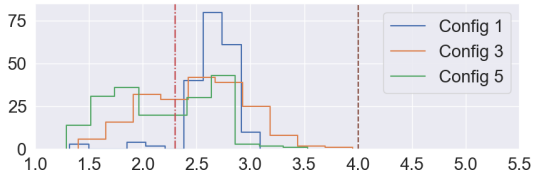


Figure 5.11: Experiment **2**, ShieldNN OFF

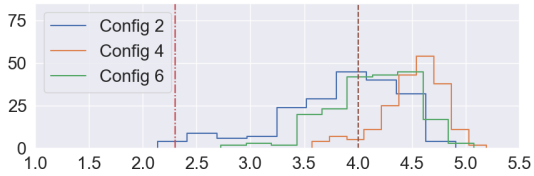


Figure 5.12: Experiment **2**, ShieldNN ON

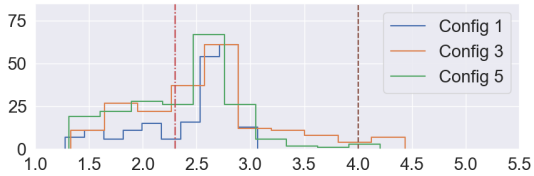


Figure 5.13: Experiment **3A**, ShieldNN OFF

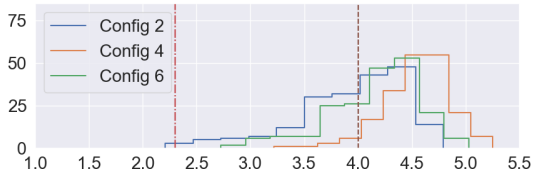


Figure 5.14: Experiment **3A**, ShieldNN ON

Figure 5.15: Distributions of distance-to-obstacles for experiments 2 & 3, with and without ShieldNN.

Part 3-A: We use the same setup and metrics as in Experiment 2, but we perturb the locations of the spawned obstacles by a Gaussian distribution $\mathcal{N}(0,\ 1.5)$m in the lateral and longitudinal directions. Fig. 5.13 and 5.14 show that despite this obstacle perturbation, ShieldNN is still able to maintain a safe distance between the vehicle and the obstacles whereas this is not the case when ShieldNN is OFF. Table 5.1 shows an overall increase of obstacle hit rate and a decrease in track completion rate when ShieldNN is OFF compared to the previous experiment. This is expected, as the PPO algorithm is trained with the obstacles spawned at locations with a different distribution than the one used in testing. However, ShieldNN continues to demonstrate its performance and safety guarantees by having almost 100% track completion rate and almost 0% obstacle hit rate.

Part 3-B: This experiment is essentially an evaluation of the ability (or not) of RL agents equipped with

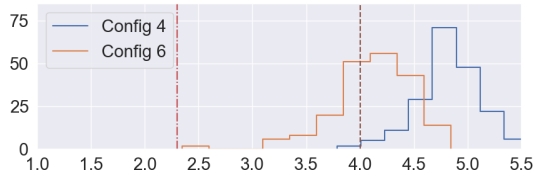


Figure 5.16: Results of Experiment 3-B, distributions of performance metrics within a completely novel environment relative to training.

ShieldNN to generalize to novel environments. To evaluate performance in this setting, a transfer learning task is implemented where the pretrained Agents 2 and 3 are then *retrained* for 500 episodes in the new environment (compare to 6000 training episodes for the original experiments). The new environment is a city road surrounded by buildings, as opposed to the urban highway environment used in the original training. This modification substantially shifts the distribution of the camera feed input.

Fig. 5.16 shows the results for Configurations 4 and 6; recall from Table 5.1 in the main text that these configurations represent when (re)training is conducted with ShieldNN OFF and ON, respectively. Note that configuration 2 – where the original training was done in an environment with no obstacles – could not successfully complete the track during re-training for 500 episodes nor in testing, and is thus not included in the figure. Observe that in both configurations, the agent is still able to avoid obstacles for the 200 number of test episodes. Furthermore, configuration 6 (both retraining and testing with ShieldNN ON), the agent appears to behave more conservatively with respect to obstacle avoidance. In conclusion, ShieldNN still achieves the desired safety distance on average, and achieving exactly zero obstacle hitting rates in both cases; it also achieves track completions of 98% and 97% respectively.

5.5.4 Experiment 4: Multiple Obstacles

In order to study the effectiveness of ShieldNN in providing safety in an environment where the vehicle can encounter multiple obstacles at the same time, we run 100 test scenarios in which we spawn multiple obstacles at random positions that are close to each other along the path of the vehicle. We compare between four variants based on a pretrained RL agent from Experiment 1 (Agent 2): 1) No filter, 2) SOSA filter; filtering PPO steering controls based on the closest obstacle to the vehicle. 3) MOSA; filtering PPO steering controls based on all detected obstacles. 4) MOSA plus penalizing the RL reward function with the unsafe action flag. This should disincentivize the RL agent from generating control actions that make the vehicle reach a state where MOSA cannot output a steering control action that is safe for all obstacles. We retrain the RL agent for 2000 episodes on scenarios that are different from the test scenarios. Then, we run the test scenarios with

	OHR%	TC%
No Filter	90	66
SOSA	22	73
MOSA (Before Retraining)	14	65
MOSA (After Retraining)	4	64

Table 5.2: Experiment 4: Multiple Obstacles Scenarios. SOSA, single-obstacle safe action; MOSA, multiple-obstacle safe action; OHR, obstacle hit rate; TC, track completion

MOSA ON after retraining. As shown in Table 5.2, using the naive approach (SOSA) reduces the obstacle hit rate significantly by 75% compared to the case when no filter is applied. Applying the proposed MOSA algorithm further reduces the obstacle hit rate by 36%. The trained RL agent with MOSA applied had the lowest obstacle hit rate of 4%. This shows that the RL agent is able to learn to avoid driving the vehicle into states where no safe action exists for all obstacles. Training the RL agent for more episodes does not improve obstacle hit rate results further. We notice that track completion percentage is relatively low for all test runs which is expected due to the fact that trying to avoid hitting multiple obstacles spawned randomly in the environment will cause the vehicle to move out of bounds of the track since ShieldNN only filters the steering control input.

5.6 Discussion

Question 1: Why is ShieldNN is different from other CBFs? The main distinction between ShieldNN and other CBF approaches can be summarized succinctly: ShieldNN does not design a controller at all, much less a *single* safe controller. Rather, ShieldNN designs one NN component that can be used to enforce safety *on any* controller, no matter how that controller was designed. This architecture gives ShieldNN two unique capabilities: (i) ShieldNN can be immediately applied to a black-box controller; and (ii) ShieldNN can be employed *during* RL training of a controller, a situation where the (learned) controller is constantly changing. In particular, optimization-based approaches — that encode the CBF as a constraint in the numerical optimization problem — cannot directly be used for the setup in the experimental section where the controller processes data collected from cameras and LiDARs to train a neural network.

Question 2: Since ShieldNN is based on the simple KBM model, will ShieldNN still work for real-world scenarios? We address this concern in two ways.

1) Several empirical studies have evaluated the effectiveness of designing controllers for actual four-wheeled vehicles, using only the KBM as a model for their dynamics [20]. By means of experimental data collected

from vehicles on a real-world test track, this reference showed evidence that the KBM is actually a viable approximation to real-vehicle dynamics with regard to controller design.

2) We validated our approach using CARLA, a simulator that simulates a much higher-fidelity dynamical model than the KBM. Even with direct parameter matching between the KBM and the CARLA vehicle, our simulation results show that ShieldNN was almost 100% effective at avoiding collisions in case of encountering one obstacle at a time and up to 96% effective at avoiding collisions with multiple obstacles at a time.

Our results do not provide the same evidence of correctness for more complicated, non-KBM dynamical models. However, we view this as a problem for future work: having established a functioning ShieldNN filter for the KBM, we can in the future focus on bounding the errors between the KBM and more complicated dynamical models, thereby obtaining more general evidence of correctness (and with some hope of success, based on points 1 and 2 above).

Question 3: Are there any side effects of ShieldNN? In our experiments, applying ShieldNN during training had the side effect of creating a higher curb hitting rate during both training and testing, as compared to the case when the agent was trained with ShieldNN OFF. In particular, after training for 6000 episodes, the curb hitting rate for agent 2 went from 100% down to 8%. However for agent 3 it went from 100% down to 30%. This is due to the fact that ShieldNN forces the vehicle to steer away from facing an obstacle which, in turn, increases the probability of hitting one of the sides of the road. This side effect suggests the possibility for future research in generalizing ShieldNN to provide safety guarantees against hitting environment boundaries as well.

Chapter 6

Conclusion

In this thesis, we have explored the impact of cyber-security, wireless connectivity, and neural network based control on the safety of autonomous systems. We have identified and significant challenges and proposed solutions to enhance the safety and security of these systems. Our research provides insights into mitigating cyber-security threats, ensuring safety in wirelessly connected autonomous systems, and designing provably safe feedback controllers for data-trained autonomous vehicles.

The first part of the thesis focused on enhancing autonomous vehicle (AV) security and safety amidst increasing cyber-security threats. We addressed the implications of applying security techniques on AV control systems, emphasizing the need to adapt low-level controllers to maintain system safety properties. Additionally, we developed techniques to handle safety issues arising from sensor spoofing attacks, aiming to infer attackers' intentions and recover compromised systems effectively.

The second part delved into the safety aspects of wireless connected autonomous systems, recognizing the potential benefits of wireless connectivity in enhancing system performance. However, the lack of provable safety guarantees in dynamic environments poses a significant challenge. Our research aimed to bridge this gap by exploring methodologies to predict non-stationary wireless channels, compute worst-case performance bounds, and plan safe trajectories for mobile vehicles, considering the interdependence between mobility, wireless communication, and safety.

Lastly, the third part of the thesis focused on ensuring the safety of NN-controlled autonomous systems. We introduced ShieldNN, an algorithm designed to produce safe data-trained feedback controllers for autonomous vehicles. By incorporating a unique architecture that corrects unsafe control actions in real-time, ShieldNN offers a promising approach to enhance the safety of NN-controlled systems.

6.1 Future Work

While our research has addressed multiple aspects of the safety and security challenges in autonomous systems, several avenues for future work remain open:

Optimization of Cyber-Security Techniques: Further research is imperative to optimize cyber-security techniques to minimize runtime overheads while maintaining system performance, especially in the context of autonomous systems. Recent studies have highlighted the significant impact of cyber-security measures on the performance of autonomous systems [128]. Exploring lightweight security mechanisms tailored specifically for autonomous systems could mitigate the impact on control performance. For instance, techniques such as lightweight encryption algorithms and efficient anomaly detection methods can help improve the security posture of autonomous systems without introducing significant computational overhead [129].

Enhancing Wireless Safety Guarantees: Future research efforts should concentrate on developing methodologies to provide provable safety guarantees for wirelessly connected autonomous systems operating in dynamic environments. The unpredictable nature of wireless channels poses significant challenges to ensuring safe operation, particularly in scenarios where multiple autonomous agents interact wirelessly. This involves predicting non-stationary wireless channels more accurately and efficiently computing worst-case performance bounds. The adoption of state-of-the-art machine learning-based channel modeling techniques, offer promising avenues for improving the accuracy of wireless channel predictions [130]. Furthermore, the integration of formal methods and control techniques, such as Control Barrier Functions (CBFs), can provide mathematical guarantees of safety for wirelessly connected autonomous systems.

Advancements in Neural Network Control: Continued exploration into enhancing the safety and reliability of Neural Network (NN)-controlled autonomous systems is crucial for their widespread adoption. Recent developments in NN-based control methods have shown promising results in improving the safety of autonomous systems. Extensions to ShieldNN and its integration into various autonomous platforms with different dynamical models could lead to more robust and dependable autonomous systems. Additionally, research efforts should focus on developing techniques to quantify the uncertainty associated with NN-based control decisions and incorporate them into the decision-making process. This can help improve the transparency and reliability of NN-controlled autonomous systems, thereby enhancing their safety and trustworthiness in real-world applications.

Safety of LLM based robots: The integration of Large Language Models (LLMs) into robotics systems presents both promising opportunities and significant safety challenges. While LLMs offer advanced capabilities in natural language understanding and generation, their deployment in robotic applications introduces

concerns regarding safety, reliability, and ethical considerations. Recent studies have highlighted potential risks associated with LLMs in robotics, such as unintended behaviors, decision-making uncertainties, and ethical dilemmas in human-robot interaction scenarios [131]. These concerns emphasize the importance of developing robust safety mechanisms to ensure the safe operation of LLM-equipped robots in real-world environments.

Future research directions in the intersection of LLMs and robotics should focus on enhancing the interpretability and transparency of LLM-based decision-making processes. Interpretable AI techniques, such as Explainable AI (XAI), can provide insights into the inner workings of LLMs, enabling users to understand and trust their behavior in critical situations [132]. By integrating XAI techniques with LLMs in robotics, researchers can enhance the explainability of robotic systems' actions, facilitating human oversight and intervention when necessary to ensure safety.

Additionally, the development of hybrid approaches that combine LLMs with traditional control methods and formal verification techniques can enhance the safety and reliability of robotic systems. By leveraging the complementary strengths of LLMs and formal methods, researchers can design robust control policies that guarantee safety while leveraging the expressive power of LLMs for complex decision-making tasks [133]. These hybrid approaches can provide a balance between autonomy and safety, enabling LLM-equipped robots to operate effectively in dynamic and uncertain environments while minimizing the risk of unsafe behaviors.

Furthermore, efforts to establish standardized safety benchmarks and evaluation metrics specific to LLM-equipped robotic systems are essential for assessing and comparing their safety performance. By developing standardized testing protocols and evaluation frameworks, researchers can systematically evaluate the safety and reliability of LLM-equipped robots across different applications and deployment scenarios. These efforts will facilitate the adoption of LLMs in robotics while ensuring that safety considerations remain paramount in their design and implementation.

Addressing safety concerns associated with LLMs in robotics requires interdisciplinary research efforts that integrate formal methods, control techniques, explainable AI, and standardized evaluation frameworks. By developing robust safety mechanisms and transparent decision-making processes, researchers can harness the potential of LLMs to enhance the capabilities of robotic systems while ensuring their safe and ethical deployment in real-world environments.

Bibliography

[1] Michele Co, Jack W. Davidson, Jason D. Hiser, John C. Knight, Anh Nguyen-Tuong, Westley Weimer, Jonathan Burket, Gregory L. Frazier, Tiffany M. Frazier, Bruno Dutertre, Ian Mason, Natarajan Shankar, and Stephanie Forrest. Double helix and raven: A system for cyber fault tolerance and recovery. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, CISRC '16, pages 17:1–17:4, New York, NY, USA, 2016. ACM.

[2] Alan Kim, Brandon Wampler, James Goppert, Inseok Hwang, and Hal Aldridge. Cyber attack vulnerabilities analysis for unmanned aerial vehicles. In *Infotech@ Aerospace 2012*, page 2438. 2012.

[3] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European symposium on security and privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[4] Rachid Ait Maalem Lahcen, Bruce Caulkins, Ram Mohapatra, and Manish Kumar. Review and insight on the behavioral aspects of cybersecurity. *Cybersecurity*, 3:1–18, 2020.

[5] Megha Agrawal, Jianying Zhou, and Donghoon Chang. A survey on lightweight authenticated encryption and challenges for securing industrial iot. *Security and privacy trends in the industrial internet of things*, pages 71–94, 2019.

[6] Farhan A Qazi. Study of zero trust architecture for applications and network security. In *2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, pages 111–116. IEEE, 2022.

[7] Devki Nandan Jha, Saurabh Garg, Prem Prakash Jayaraman, Rajkumar Buyya, Zheng Li, Graham Morgan, and Rajiv Ranjan. A study on the evaluation of hpc microservices in containerized environment. *Concurrency and Computation: Practice and Experience*, 33(7):1–1, 2021.

[8] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[9] Ralph Langner. How to kill a centrifuge, Nov 2013.

[10] Andy Greenberg. Hackers remotely kill a jeep on the highway with me in it, Jul 2015.

[11] Scott Peterson and Payam Faramarzi. Exclusive: Iran hijacked us drone, says iranian engineer. *The Christian Science Monitor*, Dec 2011.

[12] IEEE Spectrum. Protecting gps from spoofers is critical to the future of navigation. *IEEE Spectrum: Technology, Engineering, and Science News*, Jul 2016.

[13] Todd E Humphreys, Brent M Ledvina, Mark L Psiaki, Brady W O'Hanlon, and Paul M Kintner Jr. Assessing the spoofing threat: Development of a portable gps civilian spoofer. In *Proceedings of the ION GNSS international technical meeting of the satellite division*, volume 55, page 56, 2008.

[14] Vicente Milanés, Steven E Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. Cooperative adaptive cruise control in real traffic situations. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):296–305, 2014.

[15] Valerio Turri, Bart Besselink, and Karl H Johansson. Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *IEEE Transactions on Control Systems Technology*, 25(1):12–28, 2017.

[16] Kevin Driscoll, Brendan Hall, Phil Koopman, Justin Ray, and Mike DeWalt. Data network evaluation criteria handbook. DOT/FAA/AR-09/24, June 2009.

[17] Timothy Wang, Romain Jobredeaux, Marc Pantel, Pierre-Loic Garoche, Eric Feron, and Didier Henrion. Credible autocoding of convex optimization algorithms. *Optimization and Engineering*, 17(4):781–812, 2016.

[18] Petter Nilsson, Omar Hussien, Ayca Balkan, Yuxiao Chen, Aaron D Ames, Jessy W Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2016.

[19] Yushan Chen, Xu Chu Ding, and Calin Belta. Synthesis of distributed control and communication schemes from global ltl specifications. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 2718–2723. IEEE, 2011.

[20] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[21] Benjamin A. Kuperman, Carla E. Brodley, Hilmi Ozdoganoglu, T. N. Vijaykumar, and Ankit Jalote. Detection and prevention of stack buffer overflow attacks. *Commun. ACM*, 48(11):50–56, November 2005.

[22] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pages 585–590, Nov 2012.

[23] 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*, 33(1):80–86, Jan 2013.

[24] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. Control-flow integrity principles, implementations, and applications. *ACM Trans. Inf. Syst. Secur.*, 13(1):4:1–4:40, November 2009.

[25] Jun Xu, Z. Kalbarczyk, and R.K. Iyer. Transparent runtime randomization for security. In *Proceedings of the 22nd International Symposium on Reliable Distributed Systems*, pages 260–269, Oct. 2003.

[26] Jason D. Hiser, Anh Nguyen-Tuong, Michele Co, Jack W. Davidson, and Matthew Hall. ILR: Where'd my gadgets go? *IEEE Symposium on Security & Privacy*, pages 571–585, May 2012.

[27] Theuns Verwoerd and Ray Hunt. Intrusion detection techniques and approaches. *Computer Communications*, 25(15):1356 – 1365, 2002.

[28] J. McHugh, A. Christie, and J. Allen. Defending yourself: The role of intrusion detection systems. *IEEE Software*, 17(5):42–51, Sept 2000.

[29] O. Sukwong, H. Kim, and J. Hoe. Commercial antivirus software effectiveness: An empirical study. *Computer*, 44(3):63–70, March 2011.

[30] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont. An artificial immune system architecture for computer security applications. *IEEE Transactions on Evolutionary Computation*, 6(3):252–280, Jun 2002.

[31] James P Farwell and Rafal Rohozinski. Stuxnet and the future of cyber war. *Survival*, 53(1):23–40, 2011.

[32] Hackers remotely kill a jeep on the highway - with me in it. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[33] Aviva Hope Rutkin. Spoofers Use Fake GPS Signals to Knock a Yacht Off Course., 2014. MIT Technology Review, http://www.technologyreview.com/news/517686/spoofers-use-fake-gps-signals-to-knock-a-yacht-off-course.

[34] Iran-u.s. rq-170 incident. https://en.wikipedia.org/wiki/Iran-U.S._RQ-170_incident.

[35] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G.J. Pappas. Robustness of attack-resilient state estimators. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, pages 163–174, 2014.

[36] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions onAutomatic Control*, 58(11):2715–2729, 2013.

[37] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control*, 59(6):1454–1467, 2014.

[38] Nicola Bezzo, James Weimer, Miroslav Pajic, Oleg Sokolsky, George J Pappas, and Insup Lee. Attack resilient state estimation for autonomous robotic systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 3692–3698. IEEE, 2014.

[39] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. Attack models and scenarios for networked control systems. In *Proceedings of the 1st international conference on High Confidence Networked Systems*, HiCoNS '12, pages 55–64, 2012.

[40] Miroslav Pajic, Nicola Bezzo, James Weimer, Rajeev Alur, Rahul Mangharam, Nathan Michael, George J Pappas, Oleg Sokolsky, Paulo Tabuada, Stephanie Weirich, and Insup Lee. Towards synthesis of platform-aware attack-resilient control systems. In *Proceedings of the 2nd ACM international conference on High confidence networked systems (HiCoNS)*, 2013.

[41] High-assurance cyber military systems. http://www.darpa.mil/Our_Work/I2O/Programs/High-Assurance_Cyber_Military_Systems_(HACMS).aspx.

[42] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control*, 59(6):1454–1467, 2014.

[43] Radoslav Ivanov, Miroslav Pajic, and Insup Lee. Attack-resilient sensor fusion. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 54. European Design and Automation Association, 2014.

[44] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J Pappas. Robustness of attack-resilient state estimators. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*, pages 163–174. IEEE Computer Society, 2014.

[45] Md Masud Rana. Attack resilient wireless sensor networks for smart electric vehicles. *IEEE Sensors Letters*, 1(2):1–4, 2017.

[46] Yasser Shoukry, Pierluigi Nuzzo, Nicola Bezzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, and Paulo Tabuada. Secure state reconstruction in differentially flat systems under sensor attacks using satisfiability modulo theory solving. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 3804–3809. IEEE, 2015.

[47] Man-Ki Yoon, Bo Liu, Naira Hovakimyan, and Lui Sha. Virtualdrone: virtual sensing, actuation, and communication for attack-resilient unmanned aerial systems. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pages 143–154. ACM, 2017.

[48] Kaveh Paridari, Niamh O'Mahony, Alie El-Din Mady, Rohan Chabukswar, Menouer Boubekeur, and Henrik Sandberg. A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration. *Proceedings of the IEEE*, 2017.

[49] Nicola Bezzo, James Weimer, Yanwei Du, Oleg Sokolsky, Sang H Son, and Insup Lee. A stochastic approach for attack resilient uav motion planning. In *American Control Conference (ACC), 2016*, pages 1366–1372. IEEE, 2016.

[50] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[51] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

[52] Bernard Michini, Thomas J Walsh, Ali-Akbar Agha-Mohammadi, and Jonathan P How. Bayesian nonparametric reward learning from demonstration. *IEEE Transactions on Robotics*, 31(2):369–386, 2015.

[53] Graeme Best and Robert Fitch. Bayesian intention inference for trajectory prediction with an unknown goal destination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015*, pages 5817–5823. IEEE, 2015.

[54] Alberto Cerpa, Jennifer L Wong, Miodrag Potkonjak, and Deborah Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 414–425. ACM, 2005.

[55] Kannan Srinivasan, Mayank Jain, Jung Il Choi, Tahir Azim, Edward S Kim, Philip Levis, and Bhaskar Krishnamachari. The κ factor: inferring protocol performance using inter-link reception correlation. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 317–328. ACM, 2010.

[56] Gang Zhou, John A Stankovic, and Sang H Son. Crowded spectrum in wireless sensor networks. *IEEE EmNets*, 6, 2006.

[57] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13. ACM, 2003.

[58] Tao Liu and Alberto E Cerpa. Foresee (4c): Wireless link prediction using link features. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 294–305. IEEE, 2011.

[59] Carlo Alberto Boano, Marco Zuniga, Thiemo Voigt, Andreas Willig, and Kay Römer. The triangle metric: Fast link quality estimation for mobile wireless sensor networks. In *International Conference on Computer Communication Networks, 2010, Zurich, Switzerland*, 2010.

[60] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):34, 2012.

[61] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 51–62. ACM, 2006.

[62] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. *ACM SIGCOMM Computer Communication Review*, 41(4):159–170, 2011.

[63] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless networks*, 11(4):419–434, 2005.

[64] Alberto Cerpa, Jennifer L Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. Statistical model of lossy links in wireless sensor networks. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 81–88. IEEE, 2005.

[65] Shan Lin, Gang Zhou, Kamin Whitehouse, Yafeng Wu, John A Stankovic, and Tian He. Towards stable network performance in wireless sensor networks. In *2009 30th IEEE Real-Time Systems Symposium*, pages 227–237. IEEE, 2009.

[66] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM, 2003.

[67] Károly Farkas, Theus Hossmann, Lukas Ruf, and Bernhard Plattner. Pattern matching based link quality prediction in wireless mobile ad hoc networks. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 239–246. ACM, 2006.

[68] Kyu-Han Kim and Kang G Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 38–49. ACM, 2006.

[69] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pages 1–14. ACM, 2009.

[70] Yanjun Li, Jiming Chen, Ruizhong Lin, and Zhi Wang. A reliable routing protocol design for wireless sensor networks. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[71] Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. Predicting link quality using supervised learning in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(3):71–83, 2007.

[72] Marta Kwiatkowska, Gethin Norman, and David Parker. *PRISM 4.0: Verification of Probabilistic Real-Time Systems*, pages 585–591. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[73] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.*, 36(4):40–45, March 2009.

[74] Cameron Nowzari and Jorge Cortés. Distributed event-triggered coordination for average consensus on weight-balanced digraphs. *Automatica*, 68:237–244, 2016.

[75] Eloy Garcia and Panos J Antsaklis. Model-based event-triggered control for systems with quantization and time-varying network delays. *IEEE Transactions on Automatic Control*, 58(2):422–434, 2013.

[76] Li Wang, Aaron D Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2659–2664. IEEE, 2016.

[77] Li Wang, Aaron Ames, and Magnus Egerstedt. Safety barrier certificates for heterogeneous multi-robot systems. In *American Control Conference (ACC), 2016*, pages 5213–5218. IEEE, 2016.

[78] Dimitra Panagou, Matthew Turpin, and Vijay Kumar. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6757–6762. IEEE, 2014.

[79] Elzbieta Roszkowska and Ida Goral. Correct-by-construction distributed control for multi-vehicle transport systems. In *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pages 156–161. IEEE, 2013.

[80] Pedro Fernandes and Urbano Nunes. Platooning with ivc-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):91–106, 2012.

[81] Jeroen Ploeg, Elham Semsar-Kazerooni, Guido Lijster, Nathan Van De Wouw, and Henk Nijmeijer. Graceful degradation of CACC performance subject to unreliable wireless communication. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2013.

[82] Rafael Rodrigues da Silva and Hai Lin. Safety Certified Cooperative Adaptive Cruise Control under Unreliable Inter-vehicle Communications. 2016.

[83] Pangwei Wang, Yunpeng Wang, Guizhen Yu, and Tieqiao Tang. An improved cooperative adaptive cruise control (CACC) algorithm considering invalid communication. *Chinese Journal of Mechanical Engineering*, 27(3):468–474, 2014.

[84] Nathan Van De Wouw, W P Maurice H Heemels, Henk Nijmeijer, and Sinan Onc. String Stability of Interconnected Vehicles Under Communication Constraints. *IEEE Conference on Decision and Control*, pages 2459–2464, 2012.

[85] William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2067–2069, 2018.

[86] Anqi Liu, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue. Robust regression for safe exploration in control. *arXiv preprint arXiv:1906.05819*, 2019.

[87] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[88] Patricia Pauli, Anne Koch, Julian Berberich, and Frank Allgöwer. Training robust neural networks using lipschitz bounds. *arXiv preprint arXiv:2005.02929*, 2020.

[89] Chris Gaskett. Reinforcement learning under circumstances beyond its control. 2003.

[90] Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes. *arXiv preprint arXiv:1205.4810*, 2012.

[91] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4312–4320, 2016.

[92] Lu Wen, Jingliang Duan, Shengbo Eben Li, Shaobing Xu, and Huei Peng. Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization. *arXiv preprint arXiv:2003.01303*, 2020.

[93] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, 2017.

[94] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[95] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in neural information processing systems*, pages 8092–8101, 2018.

[96] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018.

[97] Anayo K Akametalu, Jaime F Fisac, Jeremy H Gillula, Shahab Kaynama, Melanie N Zeilinger, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431. IEEE, 2014.

[98] Vijay Govindarajan, Katherine Driggs-Campbell, and Ruzena Bajcsy. Data-driven reachability analysis for human-in-the-loop systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2617–2622. IEEE, 2017.

[99] Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[100] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[101] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[102] Kim P Wabersich and Melanie N Zeilinger. Scalable synthesis of safety certificates from data with application to learning-based control. In *2018 European Control Conference (ECC)*, pages 1691–1697. IEEE, 2018.

[103] Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio Vela. Synthesis of control barrier functions using a supervised machine learning approach. *arXiv preprint arXiv:2003.04950*, 2020.

[104] Andrew J Taylor, Andrew Singletary, Yisong Yue, and Aaron D Ames. A control barrier perspective on episodic learning via projection-to-state safety. *arXiv preprint arXiv:2003.08028*, 2020.

[105] Richard Cheng, Mohammad Javad Khojasteh, Aaron D Ames, and Joel W Burdick. Safe multi-agent interaction through robust control barrier functions with learned uncertainties. *arXiv preprint arXiv:2004.05273*, 2020.

[106] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. *arXiv preprint arXiv:2004.03315*, 2020.

[107] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790. IEEE, 2019.

[108] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2019.

[109] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*. Springer, 2018.

[110] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, and Mykel J Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2019.

[111] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 11423–11434, 2019.

[112] Weiming Xiang, Diego Manzanas Lopez, Patrick Musau, and Taylor T Johnson. Reachable set estimation and verification for neural network models of nonlinear dynamic systems. In *Safe, Autonomous and Intelligent Vehicles*, pages 123–144. Springer, 2019.

[113] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[114] Nicola Bezzo, Kartk Mohta, Cameron Nowzari, Insup Lee, Vijay Kumar, and George J. Pappas. Online planning for energy-efficient and disturbance-aware uav operations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, pages 5027–5033. IEEE, 2016.

[115] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):36, 2008.

[116] Clearpath. Jackal (unmanned ground vehicle).

[117] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[118] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *Intelligent Vehicles Symposium*, pages 1094–1099, 2015.

[119] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[120] Robert C Daniels and Robert W Heath. An online learning framework for link adaptation in wireless networks. In *Information Theory and Applications Workshop, 2009*, pages 138–140. IEEE, 2009.

[121] CAMP Vehicle Safety Communications Consortium et al. Us dot. vehicle safety communications project - final report. *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC*, 2006.

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

[123] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8), 2016.

[124] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. Robustness of Control Barrier Functions for Safety Critical Control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.

[125] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[126] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[127] Antonin Raffin, Ashley Hill, Kalifou René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. *arXiv preprint arXiv:1901.08651*, 2019.

[128] Michael Mylrea and Sri Nikhil Gupta Gourisetti. Cybersecurity and optimization in smart "autonomous" buildings. *Autonomy and Artificial Intelligence: A Threat or Savior?*, pages 263–294, 2017.

[129] Jie Cui, Yaning Chen, Hong Zhong, Debiao He, Lu Wei, Irina Bolodurina, and Lu Liu. Lightweight encryption and authentication for controller area network of autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 2023.

[130] Jing-Ling Wang, Yun-Ruei Li, Abebe Belay Adege, Li-Chun Wang, Shiann-Shiun Jeng, and Jen-Yeu Chen. Machine learning based rapid 3d channel modeling for uav communication networks. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–5. IEEE, 2019.

[131] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. Large language models for human-robot interaction: A review. *Biomimetic Intelligence and Robotics*, page 100131, 2023.

[132] Neil C Janwani, Ersin Daş, Thomas Touma, Skylar X Wei, Tamas G Molnar, and Joel W Burdick. A learning-based framework for safe human-robot collaboration with multiple backup control barrier functions. *arXiv preprint arXiv:2310.05865*, 2023.

[133] David Smith Sundarsingh, Jay Bhagiya, Jeel Chatrola, Adnane Saoud, Pushpak Jagtap, et al. Scalable distributed controller synthesis for multi-agent systems using barrier functions and symbolic control. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6436–6441. IEEE, 2023.