UNIVERSITY OF CALIFORNIA,
IRVINE

Security Challenges and Defense Opportunities of Connected and Autonomous Vehicle Systems in the Physical World

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Junjie Shen

Dissertation Committee:
Assistant Professor Alfred Chen, Chair
Professor Gene Tsudik
Associate Professor Marco Levorato
Assistant Professor Joshua Garcia
Associate Professor Mohammad Al Faruque

2022

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciation to my advisor Prof. Alfred Chen. Alfred has not only provided me guidance, continous support, and encouragement over my Ph.D. journey, but also set me a perfect example of an independent, diligent, and respectable researcher. I remember the first time when I chatted with him, I was facinated by his visionary plan in Autonomous Driving security research, and that was the moment that I decided to change research direction and join his lab. Fast forward four years today, I am still extremely grateful for his willingness to accept a Ph.D. student without much security background like me. Such an open-mindedness inspired me not just in my research but also in my life as well. I am very fortunate to have him being my advisor.

Next, I want to thank the other members of my dissertation committee, Prof. Gene Tsudik, Prof. Marco Levorato, Prof. Joshua Garcia, and Prof. Mohammad Al Faruque for their valuable feedback and suggestions. Also, special thanks to my intern mentors Dr. Zhisheng Hu, Dr. Shengjian Guo, and Prof. Kang Li at Baidu Security for their valuable guidance.

During my Ph.D., I worked closely with my labmates Takami Sato, Ningfei Wang, Ziwen Wan, Yunpeng Luo, Jun Yeon Won, and Jongho Lee. We shared cherishable memories in research, life, food, hobbies, etc. I am also thankful to Dr. Yunhan Jack Jia, Dr. Yantao Lu, Prof. Zhou Li, Prof. Zhe Zhou, Qifan Zhang, Mingtian Tan, Dr. Hengyi Liang, Ruochen Jiao, Prof. Qi Zhu, Zeyuan Chen, Kanglan Tang, Shinan Liu, Prof. Ziming Zhao, Prof. Chunming Qiao, Zhen Yang, Prof. Yiheng Feng, Prof. Z. Morley Mao, Prof. Yang Feng, Sumaya Almanee, Dr. Rosario Cammarota, Dr. Nahid Farhady Ghalaty, Prof. Alex Veidenbaum, Prof. Alex Nicolau, Vikram Narayanan, and Prof. Anton Burtsev for valuable discussions, technical help, and collaborations. Special thanks to Dr. Gongjin Sun and Dr. Zhi Chen for their generous help and suggestions.

I also would like to thank my friends at UCI for enriching my life in the past six years. My thnanks go to Dr. Fan Yin, Dr. Zhe Wang, Shiyao Zhang, Zhanhang Liang, Jiyang Yan, Huan Chen, Dr. Aniket Shivam, Biswadip Maity, Dr. Sajjad Taheri, Jeffrey Chen, Dr. Yu Guo, Sicong Liu, Yingtong Liu, Dr. Liangjian Chen, Deying Kong, Junze Liu, along with many others.

My Ph.D. research was supported in part by NSF under grants CNS-1850533, CNS-1929771, CNS-1932351, CNS-1932464, CNS-2145493, CNS-1823262, CMMI-2054710, USDOT under grant 69A3552047138 for CARMEN UTC, and Donald Bren School of Information and Computer Sciences at UCI. Thanks to all of them for their kind support.

Finally, I want to thank my family for all their support, sacrifices and blessings. I am eternally indebted to my parents who taught me diligence and perseverance. Last but not the least, I am forever grateful to Mengya Shi, my wife and my best friend, who has been a pillar of support during the ups and downs in this journey. Her love and endless support helped me to get through difficult times and encouraged me to push forward to complete the Ph.D. degree.

VITA

Junjie Shen

EDUCATION

Doctor of Philosophy in Computer Science **2022**
University of California, Irvine *Irvine, CA*

Master of Science in Computer Engineering **2015**
North Carolina State University *Raleigh, NC*

Bachelor of Engineering in Communication Engineering **2013**
Hangzhou Dianzi University *Hangzhou, China*

EXPERIENCE

Baidu USA, Research Intern **Mar. 2021–Dec. 2021**
Security Research Lab *Sunnyvale, CA*

Qualcomm, Performance Modeling Intern **Jun. 2017–Sept. 2017**
ARM Server CPU Team *Raleigh, NC*

AMD, Research Intern **May 2015–Aug. 2015**
AMD Research *Beijing, China*

HONORS & AWARDS

Graduate Student Entrepreneur Award in Computer Science **2021**
The Beall Family Foundation

Graduate Dean's Dissertation Fellowship **2021**
UCI Graduate Division

Best Short Paper Award **2021**
Workshop on Automotive and Autonomous Vehicle Security (AutoSec)

Champion **2020**
Baidu Security AutoDriving CTF

Best Technical Poster Award **2020**
Network and Distributed System Security Symposium (NDSS)

Distinguished Poster Presentation Award **2019**
Network and Distributed System Security Symposium (NDSS)

PUBLICATIONS

Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks **2022**
Network and Distributed System Security Symposium (NDSS)

Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack **2021**
USENIX Security Symposium (USENIX Security)

End-to-end Uncertainty-based Mitigation of Adversarial Attacks to Automated Lane Centering **2021**
IEEE Intelligent Vehicles Symposium (IV)

Fooling Perception via Location: A Case of Region-of-Interest Attacks on Traffic Light Detection in Autonomous Driving **2021**
Workshop on Automotive and Autonomous Vehicle Security (AutoSec)

Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing **2020**
USENIX Security Symposium (USENIX Security)

A Comprehensive Study of Autonomous Vehicle Bugs **2020**
ACM/IEEE International Conference on Software Engineering (ICSE)

Fooling Detection Alone is Not Enough: Adversarial Attack against Multiple Object Tracking **2020**
International Conference on Learning Representations (ICLR)

LXDs: Towards Isolation of Kernel Subsystems **2019**
USENIX Annual Technical Conference (USENIX ATC)

Combining Prefetch Control and Cache Partitioning to Improve Multicore Performance **2019**
IEEE International Parallel & Distributed Processing Symposium (IPDPS)

New Opportunities for Compilers in Computer Security **2018**
Languages and Compilers for Parallel Computing (LCPC)

A Study on Deep Belief Net for Branch Prediction **2018**
IEEE Access

CAMFAS: A Compiler Approach to Mitigate Fault Attacks Via Enhanced SIMDization **2017**
Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)

ABSTRACT OF THE DISSERTATION

Security Challenges and Defense Opportunities of Connected and Autonomous Vehicle Systems in the Physical World

By

Junjie Shen

Doctor of Philosophy in Computer Science

University of California, Irvine, 2022

Assistant Professor Alfred Chen, Chair

Recently, the technologies underneath the transportation system are under rapid revolution. At the individual vehicle level, Autonomous Driving (AD) technologies are now a reality, where a wide variety of commercial and private AD vehicles are already driving on the road. At the vehicle communication level, Connected Vehicle (CV) technologies enable Vehicle-to-Everything communications and can help the vehicles and infrastructure make more informed driving/control decisions. To facilitate driving automation and traffic control, these Connected and Autonomous Vehicle (CAV) technologies are highly security-critical as errors in them can cause various road hazards and even fatal consequences. Despite being security-critical, the security research in this domain is still largely insufficient in that (1) they mostly focus on individual AI component-level without considering the impact on the CAV systems, and (2) defensive studies, which are practically more important to advance the CAV technologies, are largely under-explored.

My dissertation aims to address these two limitations in three general directions. First, I conduct security analyses with the consideration of security challenges at the CAV system level. My research focuses specifically on the security challenges imposed by CAV's reliance on physical world information, e.g., sensor attacks and physical world attacks, and

systematically study their impact on the operation of the whole CAV system. Second, I design practical defenses against existing attacks leveraging the rich physical world information available in the CAV systems. Based on the offensive and defensive studies, my dissertation demonstrates that the reliance on the physical-layer information (e.g., sensor data, road information) inevitably introduce new security challenges (e.g., localization and perception errors) for CAVs, yet, it can also provide practical defense opportunities against existing attacks. Finally, I systematize the existing research efforts in AD security in the past 5 years and identify the important scientific gaps in prior works and propose potential solution directions.

Chapter 1

Introduction

With the increasing demand for safety, mobility, and environmental protection, revolutions are already underway in the transportation industry. At the individual vehicle level, *Autonomous Driving (AD) technologies* are now a reality in our daily life, where a wide variety of commercial and private AD vehicles are already driving on the road. For example, commercial AD services, such as self-driving taxis [87, 47], buses [11, 200], trucks [43, 204], delivery vehicles [24] are already publicly available, not to mention the millions of Tesla cars [36] that are equipped with Autopilot [33]. At the vehicle communication level, *Connected Vehicle (CV) technologies* enable Vehicle-to-Infrastructure and Vehicle-to-Vehicle communications and can help the vehicles and infrastructure make more informed driving/control decisions. Recently, government agencies across the globe are competing to push for CV deployments [381, 325, 153]. Particularly, the US is one of a few early adopters that has been testing the CV applications in US cities since 2016 [381, 380, 382, 291].

To facilitate driving automation and traffic control, these technologies, i.e., the "brains" of the *Connected and Autonomous Vehicle (CAV) systems*, are highly *security-critical* as errors in them can cause various road hazards and even fatal consequences [45, 42]. Even more

severe if such errors can be deliberately triggered by malicious parties. The CAV systems are designed to take physical world information, e.g., sensor data and road information, into consideration when making safety-critical control decisions. Unfortunately, many of such information are known controllable by attackers via sensor or physical world attack vectors. Specifically, almost all CAV sensors have been shown vulnerable to sensor attacks, including GPS [364, 198, 216, 101], camera [302, 409, 283], LiDAR [302, 336, 111], IMU [372, 368], RADAR [409], and ultrasonic sensors [409]. In addition, researchers demonstrate that AI algorithms (e.g., Deep Neural Networks or DNNs) are vulnerable to the physical world, e.g., road sign patches/posters [157, 120, 420], malicious billboard [283, 226], light projection [283, 253], etc.

Despite being security-critical, the security research in this emerging domain is still largely insufficient. First, existing works [120, 157, 420] generally focus on the security of individual AI algorithms without concretely addressing the **semantic security challenges** in the AD context. Specifically, since these AI algorithms are only components of the entire AD system, it is widely recognized that such generic AI component-level vulnerabilities do not necessarily lead to system-level vulnerabilities [332, 304, 149, 209] mainly due to the large *semantic gaps*: (i) *system-to-AI semantic gap*, which needs to overcome fundamental design challenges to map successful attacks at the AI component input space (e.g., image pixel changes [352, 175]) back to the problem space or the AD system input space (e.g., adversarial stickers [157], laser shooting [111]); and (ii) *AI-to-system semantic gap*, i.e., from AI component-level attack impacts (e.g., misdetected road objects) to those at the system level (e.g., vehicle collisions). Second, the current research focuses heavily on the offensive side, leaving the defensive studies, which are practically more important to advance the CAV technologies, largely under-explored.

1.1 Contributions

1.1.1 Overview

In this dissertation, my research aims to address the aforementioned two limitations from three general directions.

1) Security analyses with the consideration of semantic security challenges in the AD systems. My research focuses specifically on the security challenges imposed by AD system's reliance on physical world information, e.g., sensor attacks and physical world attacks, and systematically study their impact on the operation of the whole AD system. In my dissertation, we analyze the security of the two essential functional modules in AD systems–*localization* and *perception*, which localize the ego-vehicle's global position on the map and perceive surrounding obstacles and driving environment, respectively. These modules interface with the external world via sensors and thus can be most directly affected by attacks that perturb the physical world information. From the security analyses, we discover three security challenges in representative localization and perception designs that rely on such attacker-controllable physical world information.

2) Exploring practical defenses against existing attacks leveraging the rich physical world information available in the CAV systems. Despite exposing vulnerabilities when used for functional purposes, many physical world information readily-available in CAV systems can actually be *repurposed for defense purposes* with high practicality. In particular, we discover that lane or vehicle detection from the camera data presents as useful defense opportunities against existing attacks against high-level AD localization and CV-based intelligent traffic signal control systems.

3) Systematization of Knowledge (SoK) in AD security to guide future research. The AD security research domain is fast growing. In recent years, increasing numbers of research

works attempted to tackle the aforementioned semantic AI security challenges in AD context. However, so far there is no comprehensive systematization of this emerging research space. In my dissertation, we perform the first SoK of the semantic AD AI security research space by taxonomizing existing research papers and identifying scientific gaps and potential solution directions to guide future security research in this emerging domain.

In essence, my dissertation demonstrates: *The reliance on the physical-layer information (e.g., sensor data, road information) inevitably introduce new security challenges (e.g., localization and perception errors) for CAVs. Yet, such physical-layer information can also provide practical defense opportunities against existing attacks.*

1.1.2 Security Challenge in Multi-Sensor Fusion based Localization in High-Level AD Systems

For high-level AD, localization is highly security and safety critical. One direct threat to it is GPS spoofing, but fortunately, AD systems today predominantly use Multi-Sensor Fusion (MSF) algorithms that are generally believed to have the potential to practically defeat GPS spoofing. However, no prior work has studied whether today's MSF algorithms are indeed sufficiently secure under GPS spoofing, especially in AD settings. Therefore, we perform the first study to fill this critical gap. As the first study, we focus on a production-grade MSF with both design and implementation level representativeness, and identify two AD-specific attack goals, off-road and wrong-way attacks.

To systematically understand the security property, we first analyze the upper-bound attack effectiveness, and discover a take-over effect that can fundamentally defeat the MSF design principle. We perform a cause analysis and find that such vulnerability only appears dynamically and non-deterministically. Leveraging this insight, we design FusionRipper, a novel and general attack that opportunistically captures and exploits take-over vulnerabilities. We

evaluate it on 6 real-world sensor traces, and find that FusionRipper can achieve at least 97% and 91.3% success rates in all traces for off-road and wrong-way attacks respectively. We also find that it is highly robust to practical factors such as spoofing inaccuracies. To improve the practicality, we further design an offline method that can effectively identify attack parameters with over 80% average success rates for both attack goals, with the cost of at most half a day.

1.1.3 Security Challenge in Traffic Light Detection in High-Level AD Systems

The perception module is the key to the security of Autonomous Driving systems. It perceives the environment through sensors to help make safe and correct driving decisions on the road. The localization module is usually considered to be independent of the perception module. However, we discover that the correctness of perception output highly depends on localization due to the widely used Region-of-Interest (ROI) design adopted in perception. Leveraging this insight, we propose an ROI attack and perform a case study in the traffic light detection in AD systems. We evaluate the ROI attack on a production-grade AD system, Baidu Apollo, under end-to-end simulation environments. We found our attack is able to make the victim a red light runner or cause denial-of-service with a 100% success rate.

1.1.4 Security Challenge in Deep Learning based Automated Lane Centering in Low-Level AD Systems

ALC systems are convenient and widely deployed today, but also highly security and safety critical. In this work, we are the first to systematically study the security of state-of-the-art deep learning based ALC systems in their designed operational domains under physical-

world adversarial attacks. We formulate the problem with a safety-critical attack goal, and a novel and domain-specific attack vector: dirty road patches. To systematically generate the attack, we adopt an optimization-based approach and overcome domain-specific design challenges such as camera frame inter-dependencies due to attack-influenced vehicle control, and the lack of objective function design for lane detection models.

We evaluate our attack on a production ALC using 80 scenarios from real-world driving traces. The results show that our attack is highly effective with over 97.5% success rates and less than 0.903 sec average success time, which is substantially lower than the average driver reaction time. To understand the safety impacts, we conduct experiments using software-in-the-loop simulation and attack trace injection in a real vehicle. The results show that our attack can cause a 100% collision rate in different scenarios, including when tested with common safety features such as automatic emergency braking.

1.1.5 Defense Opportunity for Lateral-Direction Localization Attacks on High-Level AD Systems

As shown in the FusionRipper work [335], state-of-the-art MSF algorithms are vulnerable to GPS spoofing alone due to practical factors, which can cause various road hazards such as driving off road or onto the wrong way. In this work, we perform the first systematic exploration of the novel usage of lane detection (LD) to defend against such attacks. We first systematically analyze the potentials of such a domain-specific defense opportunity, and then design a novel LD-based defense approach, LD^3, that aims at not only detecting such attacks effectively in the real time, but also safely stopping the victim in the ego lane upon detection considering the absence of onboard human drivers.

We evaluate LD^3 on real-world sensor traces and find that it can achieve effective and timely detection against existing attack with 100% true positive rates and 0% false positive rates.

Results also show that LD^3 is robust to diverse environmental conditions and is effective at steering the AD vehicle to safely stop within the current traffic lane. We implement LD^3 on two open-source high-level AD systems, Baidu Apollo and Autoware, and validate its defense capability in both simulation and the physical world in end-to-end driving. We further conduct adaptive attack evaluations and find that LD^3 is effective at bounding the deviations from reaching the attack goals in stealthy attacks and is robust to latest LD-side attack.

1.1.6 Defense Opportunity for Data Spoofing Attacks on CV-based Intelligent Traffic Signal Control Systems

CV technologies are under rapid deployment across the globe and will soon reshape our transportation systems, bringing benefits to the mobility, safety, environment, etc. Meanwhile, such technologies also attract attention from cyberattacks. Recent work [119] has shown that CV-based Intelligent Traffic Signal Control systems are vulnerable to data spoofing attacks, which can cause severe congestion effects in the intersections. To defeat such data spoofing attacks, we explore a general detection strategy for infrastructure-side CV applications by extracting the physical-layer CV states from the readily-available infrastructure-side sensors and assigning trust scores to CVs based on the distance between physical-layer and cyber-layer CV states. However, such an approach suffers from the fundamental limitation in the sensor range. To address that, we adopt the well-established traffic models from transportation domain as traffic invariants to estimate the vehicle states out of sensor range. We then selectively remove the most suspicious CVs and re-execute the CV application to confirm their impact on the attack objective.

We implement our detector for the CV-based traffic signal control and evaluate it against two representative congestion attacks. Our evaluation in industrial-grade traffic simulator

shows that the detector is quite effective at assigning trust scores and can detect attacks with at least 95% true positive rates while keeping false positive rate below 7%. To consider the effect of sensor noises, we evaluate the detector performance under various sensor noise levels and find that it is robust even under 3× normal noise level. We also systematically explore the timeliness requirements for online detection and measure the timing overhead on an embedded device. Our results show that the online detection only increases the false positive rate by 5% in the worst case while still maintaining a high true positive rate at 98.1%.

1.1.7 SoK of Semantic AI Security in Autonomous Driving

AD systems rely on AI components to make safety and correct driving decisions. Unfortunately, today's AI algorithms are known to be generally vulnerable to adversarial attacks. However, for such AI component-level vulnerabilities to be semantically impactful at the system level, it needs to address non-trivial semantic gaps both (1) from the system-level attack input spaces to those at AI component level, and (2) from AI component-level attack impacts to those at the system level. In this work, we define such research space as *semantic AI security* as opposed to generic AI security. Over the past 5 years, increasingly more research works are performed to tackle such semantic AI security challenges in AD context, which has started to show an exponential growth trend. However, to the best of our knowledge, so far there is no comprehensive systematization of this emerging research space.

In this work, we thus perform the first systematization of knowledge of such growing semantic AD AI security research space. In total, we collect and analyze 53 such papers, and systematically taxonomize them based on research aspects critical for the security field such as the attack/defense targeted AI component, attack/defense goal, attack vector, attack

knowledge, defense deployability, defense robustness, and evaluation methodologies. We summarize 6 most substantial scientific gaps observed based on quantitative comparisons both vertically among existing AD AI security works and horizontally with security works from closely-related domains. With these, we are able to provide insights and potential future directions not only at the design level, but also at the research goal, methodology, and community levels. To address the most critical scientific methodology-level gap, we take the initiative to develop an open-source, uniform, and extensible system-driven evaluation platform, named *PASS*, for the semantic AD AI security research community. We also use our implemented platform prototype to showcase the capabilities and benefits of such a platform using representative semantic AD AI attacks.

1.2 Dissertation Organization

This dissertation is structured as follows. Chapter 2 describes background and related work for CAV security studied in my dissertation research. In Chapter 3, we analyze the security of high-level AD localization and our FusionRipper attack design and evaluation. In Chapter 4, we design DRP and ROI attacks to break the lane and traffic light detection components in AD perception, respectively. Chapter 5 then presents the LD^3 defense that can provide practical defense for state-of-the-art lateral-direction localization attacks on high-level AD systems. In Chapter 6, we leverage infrastructure-side sensors to bootstrap the trust assignment of CVs and defend against CV data spoofing attacks on intelligent traffic signal control. In Chapter 7, we systematize the existing AD security research works in the past 5 years and identify the scientific gaps. Chapter 8 concludes this dissertation and discusses potential future directions.

Chapter 2

Background and Related Work

In this chapter, we describe the background and related work for the Connected and Autonomous Vehicle (CAV) security focused on in my dissertation.

2.1 Background

2.1.1 AD Systems and AI Components

AD Levels and Deployment Status

The Society of Automotive Engineers (SAE) defines 6 AD levels – Level 0 (L0) to 5 (L5) [326], whereas the level increasing, the driving is more automated. In particular, L0 refers to no automation. However, the vehicle may provide warning features such as Forward Collision Warning and Lane Departure Warning. L1 refers to driver assistance and is the minimum level of AD. In L1 vehicles, the AD system is in charge of *either* steering *or* throttling/braking. Examples of L1 features are Automated Lane Centering and Adaptive Cruise

Control. L2 means partial automation, where the AD system controls *both* steering *and* throttling/braking. Although L1 and L2 can at least partially drive the vehicle, the driver must actively monitor and be ready to take over at any circumstances. When the autonomy level goes beyond L3, the driver does not need to be attentive when the AD system is operating in its Operational Design Domains (ODDs). However, in L3, the driver is required to take over when the AD system requests to do so. None of L4 and L5 vehicles require a driver seat. The difference is that L4 AD systems can only operate in limited ODDs whereas L5 can handle all possible driving scenarios. Currently, L2 AD is already widely available and targets consumer vehicles (e.g., Tesla Autopilot [33], Cadillac Super Cruise [350], OpenPilot [294]). L4 AD is under rapid deployment targeting transportation services such as self-driving taxis [87, 47], buses [11, 200], trucks [43, 204], and delivery robots [24], with some of them already entered the commercial operation stage that charges customers [10, 49].

Overview of AD AI Components

The AD systems generally have multiple AI components, including perception, localization, prediction, and planning as the major categories, as shown in Fig. 2.1. *Perception* refers to perceiving the surrounding environments and extracting the semantic information for driving, such as road object (e.g. pedestrian, vehicles, obstacles) detection, object tracking, segmentation, lane/traffic light detection. Perception module usually takes diverse sensor data as the inputs, including camera, LiDAR, RADAR, etc. *Localization* refers to finding the position of the AD vehicle relative to a reference frame in an environment [229]. *Prediction* aims to estimate the future status (position, heading, velocity, acceleration, etc.) of surrounding objects. *Planning* aims to make trajectory-level driving decisions (e.g., cruising, stopping, lane changing, etc.) that are safe and correct (e.g., conforming to traffic rules). Besides such modular design, people also explored *end-to-end DNN models* for AD. Currently,

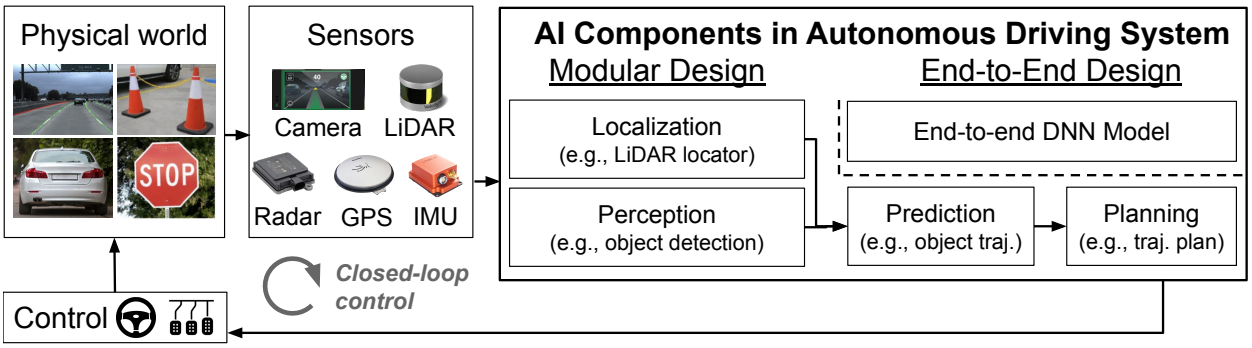

Figure 2.1: Overview of AD system designs and the roles of AD AI components.

since the modular design is easier to debug, interpret, and hard-code safety rules/measures, it is predominately adopted in industry-grade AD systems, while the end-to-end designs are only for demonstration purposes [413]. For more details, we refer the readers to recent surveys [295, 413, 229, 353].

AD Localization and Multi-Sensor Fusion

In real-world high-level (e.g., Level 4 [326]) AD system design, localization is a critical module that needs to compute global vehicle positions on the map in the real time based on positioning sensor inputs [28, 29, 31, 7, 213]. As shown in Fig. 2.2, its output is used by various other modules in the AD system, e.g., the perception module for detecting obstacles, the planning module for driving decision-making, and the control module for executing these decisions. Such direct impact on various critical decision-making steps in driving thus makes localization outputs highly security and safety critical.

To ensure safe and correct driving, AD localization needs to not only have *centimeter-level* accuracy to localize the AD vehicle at traffic lane level [319, 155, 236], but also have high robustness under various road and weather conditions [155]. Thus, Multi-Sensor Fusion (MSF) based design has become the mainstream in both academia and industry since it can fuse results from multiple independent positioning sensors, typically GPS, IMU, and LiDAR,

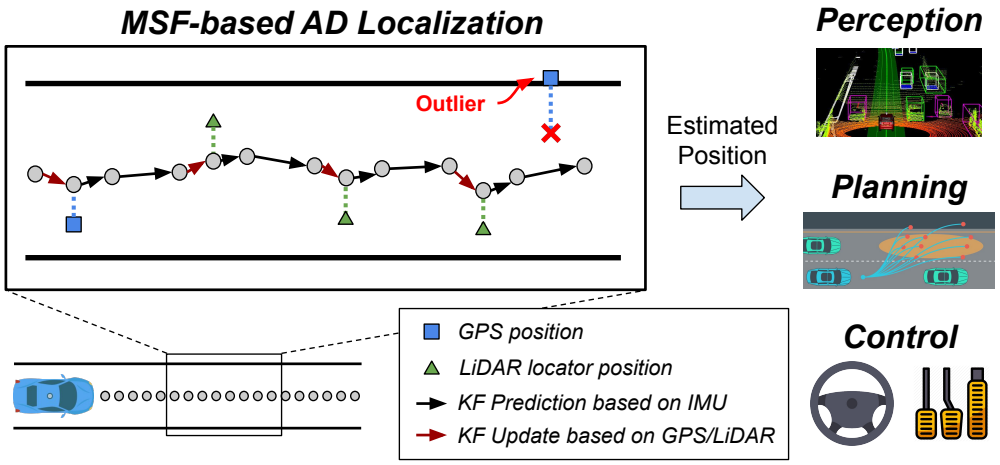

Figure 2.2: MSF-based localization and its use in high-level AD systems.

and thus produce results with overall higher accuracy and robustness [28, 29, 31, 386, 170, 346, 357, 331, 140, 340, 233, 214]. For example, modern AD-grade GPS receivers can achieve centimeter-level positioning accuracy with the error correction from ground stations [290]. However, GPS signal quality can be easily degraded due to natural phenomena such as atmosphere delays and multi-path effect [188]. LiDAR-based localization algorithms, or LiDAR locators [237, 170, 189, 102], match laser scans to pre-generated ones in a High Definition Map (HD Map) [20] in order to provide highly accurate positioning. However, the performance of such matching is susceptible to poor weather conditions such as rain or fog and the outdatedness of the HD Map. Thus, the goal of MSF is to leverage the strengths of these different sources while compensating their weaknesses.

Kalman Filter (KF) based MSF and its representativeness. Among MSF-based localization algorithms for AD systems, KF-based MSF is adopted most extensively in both academia and industry [386, 170, 357, 331, 340, 233, 214], and shown to have the state-of-the-art performance [386]. To concretely show its representativeness, we survey the MSF-based localization papers from top-tier robotics conferences [99] in the years of 2018 and 2019. As shown in Table 2.1, 14 (77.8%) of the total 18 papers adopt KF-based MSF, showing a clear predominance in today's MSF designs. Such representativeness can also be shown by the

Table 2.1: Survey of MSF-based localization designs in papers published in top-tier robotics conferences (IROS, ICRA, and RSS) [99] in the years of 2018 and 2019.

MSF Design		Papers	Percentage	
Category	Name			
KF-based	Linear KF	[305, 429, 430, 271, 152, 86, 386]	7/18 (38.9%)	14/18 (77.8%)
	Extended KF	[80, 105, 176, 418]	4/18 (22.2%)	
	Unscented KF	[106, 306, 88]	3/18 (16.7%)	
Others	Particle Filter	[415]	1/18 (5.6%)	4/18 (22.2%)
	Graph Optimization	[266, 172]	2/18 (11.1%)	
	Neural Network	[115]	1/18 (5.6%)	

fact that it is taught in all Self-Driving Car courses from Udacity [28, 29] and Coursera [31].

KF is a Bayesian filter that calculates an *optimal* state distribution with the *lowest* uncertainty from the sensor measurement distributions. In the context of AD localization, the state is composed of the vehicle's *position*, *velocity*, and *attitude* (PVA) and their *uncertainties* (or co-variance or variance matrices). Specifically, KF iteratively applies two steps: *prediction* (Eq. 2.1) and *update* (Eq. 2.2). In the k-th iteration, the inputs are the previous iteration's KF state $\hat{\mathbf{x}}_{k-1}$ and its state co-variance matrix $\hat{\mathbf{P}}_{k-1}$, which describes the state uncertainty. In the prediction, the acceleration and angular velocity from IMU are integrated in $\mathbf{F}_k$ to generate $\mathbf{x}_k$ and $\mathbf{P}_k$, which are an intermediate KF state and its co-variance. Next, the update step takes the measurement $\mathbf{z}_k$ and its uncertainty $\mathbf{R}_k$ from GPS or LiDAR locator, and first use $\mathbf{R}_k$ to calculate Kalman gain $\mathbf{K}_k$. $\mathbf{K}_k$ is then used as a weight to determine how much of the difference between $\mathbf{z}_k$ and $\mathbf{x}_k$ is updated to the new state $\hat{\mathbf{x}}_k$, and how much of $\mathbf{P}_k$ is updated to the new state co-variance $\hat{\mathbf{P}}_k$. In the equations, $\mathbf{Q}$ and $\mathbf{H}$ are typically constant matrices, with the former used for tuning the system and the latter for mapping the state space to the measurement space.

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \\ \mathbf{P}_k &= \mathbf{F}_k \hat{\mathbf{P}}_{k-1} \mathbf{F}_k^T + \mathbf{Q} \end{aligned} \tag{2.1}$$

$$
\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{x}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) \\
\hat{\mathbf{P}}_k &= \mathbf{P}_k - \mathbf{K}_k\mathbf{H}\mathbf{P}_k \\
\mathbf{K}_k &= \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}_k)^{-1}
\end{aligned}
\tag{2.2}
$$

Fig. 2.2 shows an example of the KF operations. In the prediction step, the acceleration and angular velocity from IMU are integrated in the KF to generate an intermediate state (black arrows in Fig. 2.2). In the update step, KF takes the position measurements from GPS or LiDAR locator, and updates a fraction of it to the KF state based on the uncertainties of the KF state and the measurement. A larger KF state uncertainty or a smaller measurement uncertainty will cause more updates to the KF state.

Outlier detection. To prevent KF state from being easily disrupted by occasional measurements that are too noisy in the real world, the KF update is usually bounded by an *outlier detector*. Fig. 2.2 shows an example where a GPS measurement is discarded since its position deviates too much from the KF state. Chi-squared test (Eq. 2.3) is one of the most widely used outlier detectors for KF [331, 214, 303], which considers a measurement $\mathbf{z}_k$ as an outlier if the Chi-squared test value χ_k^2 is larger than a statistical significance threshold (usually 3.841 [135]). An outlier measurement can be either discarded or partially updated.

$$
\begin{aligned}
\chi_k^2 &= (\mathbf{z}_k - \mathbf{H}\mathbf{x}_k)^T\mathbf{S}_k^{-1}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) \\
\mathbf{S}_k &= \mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}_k
\end{aligned}
\tag{2.3}
$$

Targeted MSF implementations and representativeness. In our work, the main target is an MSF design and implementation from the Baidu Apollo team, which we call *BA-MSF*. It is published in ICRA 2018 [386], a top-tier robotics conference [99], and follows the KF-based MSF design using high-end GPS, LiDAR, and IMU, with the Chi-squared test as the

outlier detector conforming to the common practice [331, 214, 303]. As described earlier, such design is the most representative in today's MSF-based AD localization (Table 2.1).

Besides its design, the implementation of BA-MSF is also highly representative in today's MSF-based AD localization: it has been tested using a large AD vehicle fleet in various challenging scenarios such as urban downtown, highways, and tunnels [386], and shown the *highest* localization accuracy (0.054 m) among *all* MSF-based localization papers (including both KF-based and non KF-based) in the top-tier robotics conferences [99] in the years of 2018 and 2019. Today, it is already adopted in Baidu Apollo [7], a production-grade AD system currently providing self-driving taxi services in China [9].

Besides BA-MSF, we also consider two other publicly-available KF-based MSFs for generality evaluations. We follow the common parameter tuning process [177] but can only reach at most 1–2 meter accuracy, which is far from the centimeter-level accuracy required by AD systems [236, 319]. Thus, in the majority of our experiments, we target BA-MSF as it is much more representative for AD systems.

DNN-based Automated Lane Centering

Fig. 2.3 shows an overview of a typical ALC system design [66, 234, 131], which operates in 3 steps:

Lane Detection (LD). Lane detection (LD) is the most critical step in an ALC system, since the driving decisions later are mainly made based on its output. Today, production ALC systems predominately use front cameras for this step [32, 33, 2, 40]. On the camera frames, an LD model is used to detect lane lines. Recently, DNN-based LD models achieve the state-of-the-art accuracy [391, 297, 223] and thus are adopted in the most performant production ALC systems today such as Tesla Autopilot [33]. Since lane line shapes do not change much across consecutive frames, recurrent DNN structure (e.g., RNN) is widely

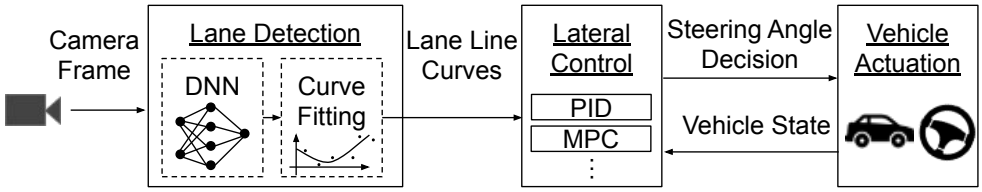

Figure 2.3: Overview of the typical ALC system design.

adopted in LD models to achieve more stable prediction [239, 428, 131]. LD models typically first predict the lane line points, and then post-process them to lane line curves using curve fitting algorithms [391, 338, 297, 168].

Before the LD model is applied, a Region of Interest (ROI) filtering is usually performed to the raw camera frame to crop the most important area out of it (i.e., the road surface with lane lines) as the model input. Such ROI area is typically around the center and much smaller than the original frame, to improve the model performance and accuracy [411].

Lateral control. This step calculates *steering angle decisions* to keep the vehicle driving at the center of the detected lane. It first computes a desired driving path, typically at the center of the detected left and right lane lines [97]. Next, a control loop mechanism, e.g., Proportional-Integral-Derivative (PID) [145] or Model Predictive Control (MPC) [322], is applied to calculate the optimal steering angle decisions that can follow the desired driving path as much as possible considering the vehicle state and physical constraints.

Vehicle actuation. This step interprets the steering angle decision into actuation commands in the form of *steering angle changes*. Here, such actuated changes are limited by a maximum value due to the physical constraints of the mechanical control units and also for driving stability and safety [97]. For example, in our experiments with a production ALC with 100 Hz control frequency, such limit is 0.25° per control step (every 10 ms) for vehicle models [39]. As detailed later in §4.2.2, such a steering limit prevents ALC systems from being affected too much from successful attack in one single LD frame, which introduces a unique challenge to our design.

Traffic Light Detection and Region-of-Interest

Traffic Light (TL) detection is an essential feature for Level-4 AD vehicles. If an AD vehicle cannot detect and recognize TLs, it may violate traffic rules resulting in some catastrophic consequences, such as running red lights and causing car accidents. Recently, as Level-2 AD companies aiming to achieve higher level driving autonomy, companies such as Tesla starts to add TL detection into their AD systems [35]. TL detection leverages Deep Neural Networks (DNNs) to detect and classify TLs in the camera images [7, 213]. For example, Baidu Apollo applies two DNNs for TL detection; one for the object detection to recognize the TL object in the image, and the other for the classification to recognize the light color [7].

Since the camera image may contain multiple TLs, it is thus necessary to identify the correct TL for the current intersection. To address this, AD systems commonly adopt a Region-of-Interest (ROI) design [7, 213], which projects the current TL in the world coordinates obtained from the High-Definition (HD) map to image coordinates based on the localization outputs (Fig. 2.4). To compensate for any localization or HD map inaccuracies, an ROI area with an empirically determined radius or height/width centered at the projected image coordinates is selected for TL detection. Such an ROI design not only helps prevent ambiguous TL detection but also reduces the computation overhead. Fig. 2.5 show an example of the ROI in Baidu Apollo. As shown, the ROI anchor box is the projected position of the TL. However, due to inaccuracies in localization and HD map, the projected position is not perfectly aligned with the actual TL. Thus, Baidu Apollo defines a larger rectangle as the ROI for TL detection (② in Fig. 2.5). After that, it crops the ROI area from the image and applies DNNs to recognize all TL bounding boxes and their colors in the ROI. Finally, the TL bounding box (③ in Fig. 2.5), which is the closest to the ROI anchor box, is selected as the TL detection result.

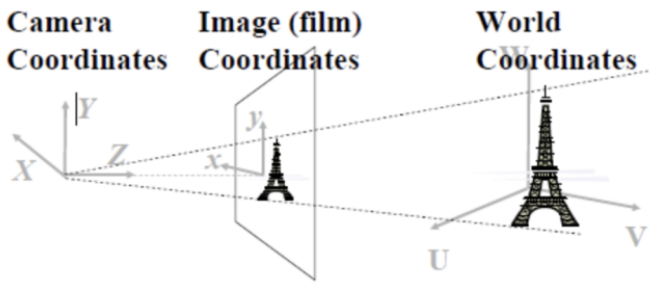

Figure 2.4: Projection from world coordinates to image coordinates.

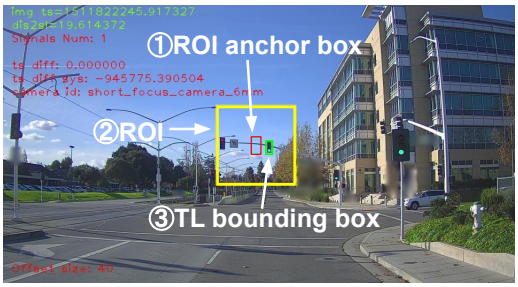

Figure 2.5: Illustration of the ROI in TL detection.

2.1.2 AD AI related Security Backgrounds

Adversarial AI Attacks and Defenses

Recent works find that AI models (e.g., DNNs) are generally vulnerable to adversarial examples, or adversarial attacks [352, 112]. Some works further explored such attacks in the physical world [120, 157, 111, 110]. With that, some defenses [407, 232] are proposed recently to defend against such attacks. Such AI attacks/defenses are directly related to AD systems due to the high reliance on AI components (§2.1.1). However, generic AI component-level vulnerabilities do not necessarily lead to system-level vulnerabilities due to the general system-to-AI and AI-to-system semantic gaps. In our SoK (Chapter 7), we thus focus on the works that address *semantic AI security* in the AD context.

GPS Spoofing and its Practicality

GPS spoofing has been a fundamental problem for civilian GPS systems due to the lack of signal authentication in the infrastructure. In GPS spoofing, the attacker transmits fabricated GPS signals with stronger power than the authentic ones, and thus causes the victim receiver to lock onto the attacker's signals and resolve positions controlled by the attacker. GPS spoofing has been proven feasible theoretically [364] and empirically [198]. So far, it has

been demonstrated on various end systems such as smartphones [414, 280], drones [164, 216], yachts [101], and recently also low-level AD vehicles such as Tesla cars [37]. Recently, a year-long investigation identified 9,883 spoofing events that affected 1,311 civilian vessel systems in Russia since 2016 [109]. Although GPS spoofers are illegal to be sold in the U.S., they can be made cheaply from commercial off-the-shelf components. For example, a low-end spoofer is as cheap as $223 [414], and higher-end ones that can simultaneously track 10+ satellites and transmit 10+ fake GPS signals only cost similar to a laptop [288, 198]. Considering such high realism, in this work, we consider it as a practical attack vector to AD localization.

2.1.3 CV-based Traffic Signal Control and Traffic Modeling Basics

CV-based Traffic Signal Control

The traffic signal control is one of the important applications of CV technologies designed to improve the transportation mobility by reducing vehicle delays in the intersection. It determines the signal timing plan based on the real-time vehicle states (including vehicle ID, location, speed, heading, etc.) periodically broadcast from the CVs in the intersection. Specifically, the CVs send their vehicle states in a standard Basic Safety Message (BSM) format, which is transmitted over the CV communication network, i.e., Dedicated Short-Range Communication (DSRC) [215] or Cellular Vehicle-to-Everything (C-V2X) [299]. Among the open efforts in CV deployment in the US [381], the Intelligent Traffic Signal System (I-SIG) [185] is the only traffic signal system designed for the mobility of general urban intersections [16]. It has been tested in real intersections in US cities and shown high effectiveness at reducing the vehicle delays [75]. Therefore, we demonstrate the effectiveness of our detector on the I-SIG system in this work.

We now illustrate some basic concepts in CV-based traffic signal control. Fig. 2.6 shows the configuration of a common major arterial intersection. As shown in the figure, the traffic

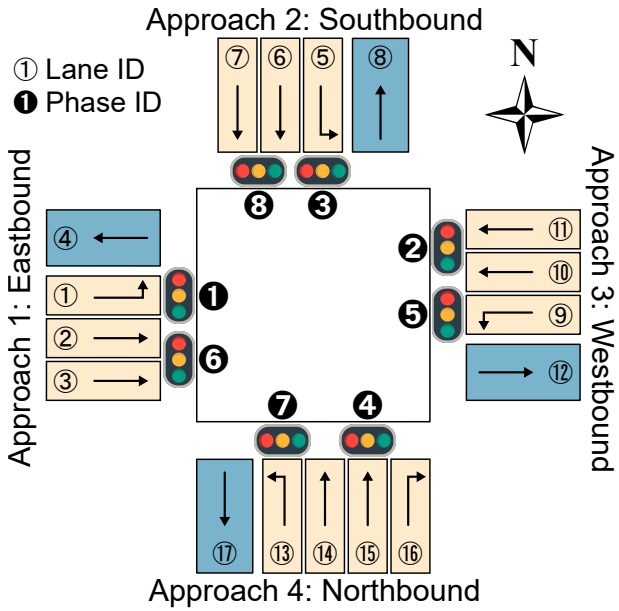

Figure 2.6: Illustration of a common major arterial intersection.

lanes are grouped by different *phases*, each with a dedicated traffic signal. As a major arterial intersection, it has two concurrent phase sequences (or *rings*) that do not interfere with each other and can be planned simultaneously. However, the phases in the same ring need to be planned sequentially due to the confliction. The 8 phases are separated into two *stages*, which are also conflicting with each other. The CV-based traffic signal controller is invoked at the end of each stage to calculate an *allocation of green lights* to the phases, which is called a *signal timing plan*. Fig. 2.7 shows a typical timing plan for 8-phase intersections. The green blocks indicate the allocated green light durations. The yellow and red blocks are two predefined durations for the yellow light and red clearance light (to accommodate for potential red light runners). The inputs to the signal controller are the latest CV states received at the end of each stage, which we call them as a CV or traffic *snapshot* in this work.

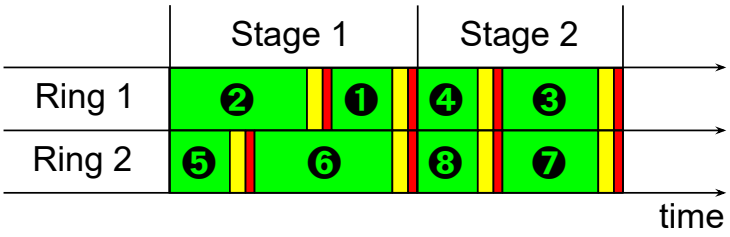

Figure 2.7: A typical signal timing plan for 8-phase intersections. The circled numbers with solid background denote the phase IDs.

Traffic Modeling

Traffic modeling is a well-established topic in transportation engineering, where people aim to precisely describe the relationships between vehicles and infrastructure with mathematical equations [268]. Traffic models can be categorized into 3 major types based on the modeling granularity. (1) *Microscopic* models, which describe the behavior of individual objects, e.g., how does an individual vehicle reacts to the actions of other vehicles. (2) *Macroscopic* models, which describe the behaviors of the aggregated traffic flow dynamics, e.g., the changes of traffic flow and density dynamically w.r.t. the road conditions. (3) *Mesoscopic* models, which stand in between microscopic and macroscopic models and are able to describe the traffic flow over time. Since traffic models are designed to model the real-world traffic or vehicle behaviors, they are widely used in the transportation domain to help design road networks, estimate the impact of transportation policies, etc.

Traffic models as the traffic invariants. Due to the attractive property of describing normal traffic behaviors, we take these traffic models as the traffic invariants for security purposes, e.g., benign vehicles will generally behave according to the traffic models. Although traffic flow statistics from the macroscopic and mesoscopic models may also be useful, the individual vehicle behaviors derived from the microscopic models are mostly relevant in the context of data spoofing detection.

Newell's car-following model. Car-following models [324, 287, 173] are the most typical

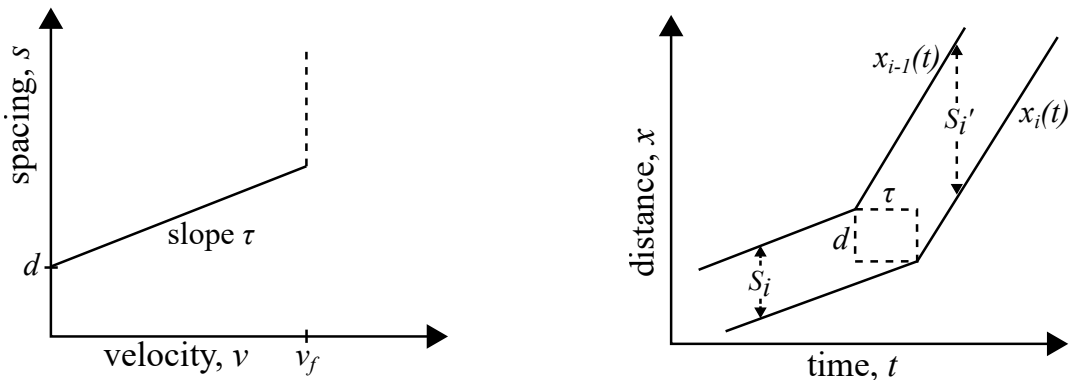

Figure 2.8: Illustration of the Newell's car-following model [287]. Left: the speed-spacing relationship for a single vehicle. Right: the leader-follower trajectory relationships.

microscopic model type that describe how should a vehicle follow its leading vehicle. The Newell's car-following model [287] is a simple but effective model, which is built upon the assumption that the follower vehicle's trajectory looks similar to its leading vehicle's trajectory, except with a time delay and a space translation. Fig. 2.8 illustrates the properties specified by the Newell's model. More formally, the model describes the following relationships:

$$\begin{aligned} s &= v \cdot \tau + d \\ x_{\text{follow}}(t+\tau) &= x_{\text{lead}}(t) - d, \end{aligned} \tag{2.4}$$

where s is the spacing that the following vehicle keeps to the leading vehicle; x_{follow} and x_{lead} are the longitudinal position offsets of the following and leading vehicles; τ and d are two empirically determined parameters describing the reaction time and stopping distance of a normal driver, which are usually set between 1.0–1.7 and 6.0–9.6, respectively [76, 398]; v_f is the free-flow speed, which is the speed limit of the road.

The intuitions behind the Newell's model are quite straightforward. The left figure in Fig. 2.8 shows that an individual vehicle will naturally try to maintain a larger spacing with the front vehicle if the speed increases. And in the right figure, the vehicle i was originally following the vehicle $i-1$ with a spacing S_i. When the leading vehicle speeds up, the follower also speeds up after a reaction time delay τ. Since now the speed is higher, the spacing maintained by

the follower also increases.

2.1.4 Congestion Attacks against CV-based Traffic Signal Control

Recent work [119] discovered two vulnerabilities in the I-SIG system that can be exploited using data spoofing attacks. The first one is in queue length estimation, where the I-SIG system estimates the number of vehicles stopping in a queue based on the farthest stopped CV in the transition periods, i.e., when the Penetration Rate (PR) is <95%. They find that such a design can be exploited by the attacker to inject a fake long queue in the estimation, which we term ***queue length attack***. The second vulnerability targets the arrival time estimation, where I-SIG estimates when will a vehicle arrive at the stop line or stop behind a queue. An attacker can exploit this by setting the CV with a slow speed to cause a late arrival time estimation, which we term ***arrival time attack***. In both attacks, the I-SIG system will allocate an *unnecessary* long green time for traffic lanes that are not busy and thus starve the other lanes that actually need prioritization. The attacks are evaluated in an industrial-grade traffic simulator named PTV Vissim [311]. Results show that the two attacks can effectively cause severe congestion effects in full deployment and transition periods of CVs, respectively.

2.2 Related Work

GPS spoofing on navigation systems. Recently, Zeng et al. [414] find that GPS spoofing can be used to stealthily deviate a victim car to an attacker-controlled destination. Later Narain et al. [280] further find that such attack also exists for a GPS/INS (Inertial Navigation System) navigation system. Compared to our work on MSF-based localization (Chapter 3), these prior works target single-source localization systems without fusion from other position

sources, such as a LiDAR locator.

Theoretical works on KF security. Existing theoretical works [345, 249, 274, 273] from the control systems domain have studied the security of KF under sensor spoofing. Compared to our work, they only study single-source KFs without any sensor fusion. Also, they focus on the theoretical aspect of the KF and assume the attacker has full access to the KF internals, e.g., KF state and uncertainties. In comparison, our work (Chapter 3) does not make such assumptions and hence is much more realistic.

Sensor attacks and defenses. Various previous works studied security problems on traditional vehicle systems [116, 95, 171], but not AD systems. Prior works discovered various sensor attack vectors on sensors related to AD systems, such as camera [302, 409, 283], LiDAR [302, 336, 111], IMU [372, 368], RADAR [409], and ultrasonic sensors [409]. However, none of them considers how to leverage these attack vectors to attack AD localization. On the defense side, researchers recently propose physical-invariant based defenses, CI [126] and SAVIOR [313], to detect sensor attacks such as GPS spoofing by cross-checking sensor measurements with system state estimations based on the physical invariants, i.e., the relationships between system states and control inputs. However, as will be shown in §5.5.2, the direct adaptation of existing physical-invariant based approach is largely limited because of the complexity of physical dynamics and much smaller attack deviation goals in the AD context. In addition, none of them has proposed attack response designs, which is especially important for AD systems. Nevertheless, such physical-invariant based attack detection methods are complimentary to our work (Chapter 5) and can be incorporated into our defense design for attack detection if the accuracy of state-estimation model can be further improved.

Attack response/recovery methods. According to a survey on the broader Cyber-Physical Systems security, existing defenses mostly focus on attack detection and very few works studied attack responses [174]. Particularly, Choi et al. [125] and Zhang et al. [416]

recently propose *attack recovery* methods, which apply similar state estimations as above to replace attacked sensors in the attack recovery period. Thus, they suffer from the same model accuracy limitations in the AD context. Moreover, they intend to maintain normal operations of the system for a short duration until the system is taken-over by the human driver, which does not exist on high-level AD vehicles when deployed commercially [87, 222]. Additionally, attack responses in high-level AD systems require more careful design on AR trajectories to safely navigate the vehicle.

AD perception security. Prior works have studied the security of AD perception related AI components, such as object detection [157, 120, 420, 111, 67], tracking [208], and end-to-end AD models [301, 362, 124, 422]. One of our works (§4.2) studies autonomy software security in production ALC. The only prior effort is from Tencent [59], but it neither attacks the designed operational domain for ALC (i.e., roads with lane lines), nor generates perturbations systematically by addressing the design challenges in §4.2.2.

Physical-world adversarial attacks. Multiple prior works have explored image-space adversarial attacks in the physical world [227, 333, 90, 108, 120, 157, 421, 420, 240]. In particular, various techniques have been designed to improve the physical-world robustness, e.g., non-printability score [333, 120, 157, 421], low-saturation colors [420], and EoT [90, 108, 120, 157, 420]. In comparison, prior efforts concentrate on image classification and object detection, while our work (§4.2) is the first to systematically design physical-world adversarial attacks on ALC, which require addressing various new and unique design challenges (§4.2.2). Specifically for TL detection, the TLs are not as accessible as other road objects (e.g., STOP signs) since they are usually installed at a high place, and thus it is unclear how the attacker can put the stickers without alerting other drivers and law enforcement. Another line of research explored physical sensor attacks to blind the camera by shooting strong lights to it [302]. However, such attacks cannot cause the victim to be a red light runner and can only lead to denial-of-service. In our work (§4.1), the ROI attack does not require any physical

perturbations to the TLs and can achieve both attack goals.

Security analyses of CV applications. Previous works [193, 397, 231, 230] have studied the security threats on the CV communication networks, such as DoS, spamming, masquerading, black hole, replay attack, which greatly damage the availability, authenticity, and confidentiality of the network. On the other side, domain-specific attacks have been demonstrated in different CV application scenarios. Chen et al. [119] show that CV-based Intelligent Traffic Signal System is highly vulnerable to congestion attacks if the attacker can compromise the OBU device. Amoozadeh et al. [81] demonstrate that message falsification attack cause significant instability in the Cooperative Adaptive Cruise Control (CACC) vehicle stream. Abdo et al. [71] perform a detailed analysis on CACC and present 4 different attack scenarios. Furthermore, Huang et al. [197] analyze the impact of falsified CV data and propose a black-box attack to the CV-based traffic signal control system.

Defenses against data spoofing in the CV context. Sun et al. [348] propose a verification scheme approach which utilizes the angle-of-arrival and frequency-of-arrival to detect spoofing attacks. Such a defense requires extra hardware and the presence of enough number of reflectors in the driving environment. Guo et al. [180] propose a collaborative intrusion detection system, which leverages the sensor data from onboard sensors of neighboring CVs. Liu et al. [250] propose a blockchain-based framework to build trust and defeat spoofing attacks in CV applications. Both works assume specific hardware or software updates in the CVs, which may require enormous efforts due to the large number of CVs. In comparison, our detector does not impose such requirements on the CVs. Instead, we reuse the readily-available infrastructure-side sensors to establish the physical root-of-trust for the detection. Hu et al. [192] build a system named CVShield based on hardware-assisted security feature (e.g., ARM Trust Zone) to prevent compromised CVs from sending falsified data, which requires vehicle-side software update and is orthogonal to our method. In our work (Chapter 6), we approach the problem from a novel angle by constructing and propagating trust

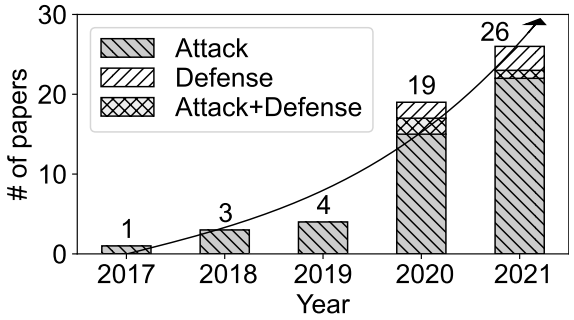

Figure 2.9: Number and trends of semantic AD AI security papers (collected with a focus on top-tier venues).

based on infrastructure-side sensors and traffic invariants, which can also be complementary to existing defenses.

Systematization of Knowledge on AD security. Before our work (Chapter 7), a few AD security-related surveys have been published [142, 312, 218, 320], but none of them focus on the emerging semantic AD AI research space. For example, Kim et al. [218] and Ren et al. [320] focus on AD-related sensor/hardware security and in-vehicle network security, instead of AD AI components. Qayyum et al. [312] and Deng et al. [142] touched upon the security of AI components in AD, but did not focus on the works that addressed the semantic AI security challenges (e.g., most of the included works are on generic AI and sensor security without studying impacts on AD AI behaviors). In fact, Deng et al. [142] considers the semantic AD AI security as a future direction. This is mainly because both of them focus on works published in 2019 and before. However, the majority (85%) and the exponential growth of semantic AD AI security works are after 2019 (Fig. 2.9). This leads to much more complete, diverse, and quantifiable observations of the current status, trends, and scientific gaps of this emerging research space, not to mention that we also take the initiative to address one of the most critical gaps (§7.5).

In the general CPS (Cyber-Physical System) area, there are also Systematization of Knowledge (SoK) papers on the security of technologies related to AD AI, for example on drones [282] that share similar controller designs, on Automatic Speech Recognition and Speaker Iden-

tification (ASR/SI) [73] that are also AI-enabled CPS, and on sensor technology [408] that provide the main inputs to AD AI. In comparison, the SoKs on drone and ASR/SI security focused on domain-specific attack vectors (e.g., ground control station channel and voice signals), which thus lead to vastly different set of systematized knowledge due to the CPS domain differences. The SoK on sensor security is complementary to ours as it does not directly consider AI component security but sensor attack is one major attack vector to AD AI.

Chapter 3

Security Challenge in AD Localization

3.1 Introduction

Today, various companies are developing high-level self-driving cars [3] such as Level-4 Autonomous Driving (AD) vehicles [326], and some of them are already providing services on public roads such as self-driving taxi from Google's Waymo One [47] and self-driving trucks from TuSimple [43]. To enable such high-level driving automation, the AD system needs to not only perform the perception of surrounding obstacles, but also *centimeter-level localization* of its own global positions on the map [236, 319]. Such localization function is highly security and safety critical in the AD context, since positioning errors can directly cause an AD vehicle to drive off road or onto a wrong way. Since in high-level AD systems the perception module is only designed for obstacle detection and the localization module is in full charge of identifying road deviations [28, 29, 31, 7, 213], even when the perception module is functioning perfectly, it cannot prevent a variety of road hazards specific to localization errors such as driving off road to hit road curbs, falling down the highway cliff, or being hit by other vehicles that fail to yield, especially when the AD vehicle is on the

wrong way. However, recent security research in AD systems concentrates on AD perception, e.g., malicious stickers on traffic signs [158, 157, 420, 111], which leaves the security of AD localization an open problem.

For outdoor localization in general, GPS is the *de facto* location source, and thus a direct threat to it is GPS spoofing, a long-existing but still unsolved security problem with practicality proven on a wide range of end systems [364, 198, 414, 280, 164, 216, 37, 101, 289], including low-autonomy AD vehicles such as Tesla cars [37]. Fortunately, to achieve robust localization, real-world high-level AD systems today predominantly use Multi-Sensor Fusion (MSF) algorithms that combine GPS input with position inputs from other sensors, typically IMU (Inertial Measurement Unit) and LiDAR (Light Detection and Ranging) [28, 386, 170, 346, 357, 331, 140, 340, 233, 214]. Since in such design GPS input alone can not dictate the localization output, it is generally believed to have the potential to practically defeat GPS spoofing [101, 414, 235, 139, 79, 201]. However, state-of-the-art MSF algorithms are mainly designed for improving accuracy and robustness, instead of security. This thus makes it largely unclear how secure they can be under GPS spoofing. Given its widespread use in AD vehicles and high importance to road safety, it is thus imperative to systematically understand this as early as possible.

To fill this critical research gap, in this work we perform the first study on the security property of MSF-based localization in AD settings. As the very first study in this direction, we focus on GPS spoofing as the attack vector since it is one of the most mature attack vectors to the MSF input sources. We focus on a production-grade MSF implementation, Baidu Apollo MSF (BA-MSF), due to its high representativeness in both design (KF-based MSF) and implementation (centimeter-level accuracy evaluated by real-world AD vehicle fleet). We consider the attack goal as using GPS spoofing to cause large *lateral* deviations in the MSF output, i.e., deviating to the left or right. This can cause the AD vehicle to drive off road or onto a wrong way, which we call *off-road attack* and *wrong-way attack* respectively.

To systematically understand the security property, we first analyze the upper-bound attack effectiveness via a dynamic blackbox analysis since BA-MSF is released in the binary form. We find that in the real-world trace, the majority (71%) of even such upper-bound attack results can only cause less than 50 cm deviation, which is far from causing either off-road or wrong-way attacks (need over 90 cm and 2.4 m respectively). This shows that MSF can indeed generally enhance the security against GPS spoofing. Interestingly, we also observe that there still exist a few upper-bound attack results that can cause over 2 meters deviations. For all of them, we find that GPS spoofing is able to cause *exponential growths* of deviations. This allows the spoofed GPS to become the dominating input source in the fusion process and eventually cause the MSF to reject other input sources, which thus *fundamentally defeats the design principle of MSF*. In this work, we call it a *take-over effect*. We then perform a cause analysis and find that this only appears when the MSF is in relatively *unconfident* periods due to a combination of dynamic and non-deterministic real-world factors such as sensor noises and algorithm inaccuracies.

Such take-over vulnerabilities are highly attractive for attackers since they can exploit the exponential deviation growths to achieve *arbitrary* deviation goals. However, as discovered earlier, the vulnerable periods are created dynamically and non-deterministically. Thus, we design *FusionRipper*, a novel and general attack that opportunistically captures and exploits take-over vulnerabilities with 2 stages: (1) *vulnerability profiling*, which measures when vulnerable periods appear, and (2) *aggressive spoofing*, which performs exponential spoofing to exploit the take-over opportunity.

We implement FusionRipper and evaluate it on 6 real-world sensor traces from Apollo and the KAIST Complex Urban dataset. Our results show that when the attack can last 2 minutes, there *always* exists a set of attack parameters for FusionRipper to achieve *at least* 97% and 91.3% success rates in *all* traces for the off-road and wrong-way attacks respectively, with less than 35 seconds success time on average. To understand the attack practicality,

we evaluate it with practical factors such as (1) spoofing inaccuracies, and (2) AD control taking effect, and find that for both cases the attack success rates are affected by less than 4%. Attack demos showing the end-to-end attack impact are available at `https://sites.google.com/view/cav-sec/fusionripper`.

In addition, we observe that the attack effectiveness is sensitive to the selection of the attack parameters. Thus, to improve the practicality, we further design an offline attack parameter profiling method that can collect effective parameters without causing obvious safety problems during such profiling to stay stealthy. Our results on real-world traces show that our method can effectively identify attack parameters with 84.2% and 80.7% success rates for off-road and wrong-way attacks respectively, with the profiling cost of at most half a day.

Considering the critical role of localization for safe and correct driving, the discovered attack against the state-of-the-art MSF algorithm requires immediate attention and defense discussion. To facilitate this, we also discuss both long-term and short-term defense directions.

In summary, this work makes the following contributions:

- We perform the first security study on MSF-based localization in high-level AD settings under GPS spoofing. We focus on a production-grade MSF with both design and implementation level representativeness, and identify two attack goals specific to the AD settings.
- We analyze the upper-bound attack effectiveness, and discover a take-over effect that can fundamentally defeat the MSF design principle. We further perform a cause analysis and find that such vulnerability only appears dynamically and non-deterministically.
- We design FusionRipper, a novel and general attack that opportunistically captures and exploits the take-over vulnerability we discover. We evaluate it on 6 real-world

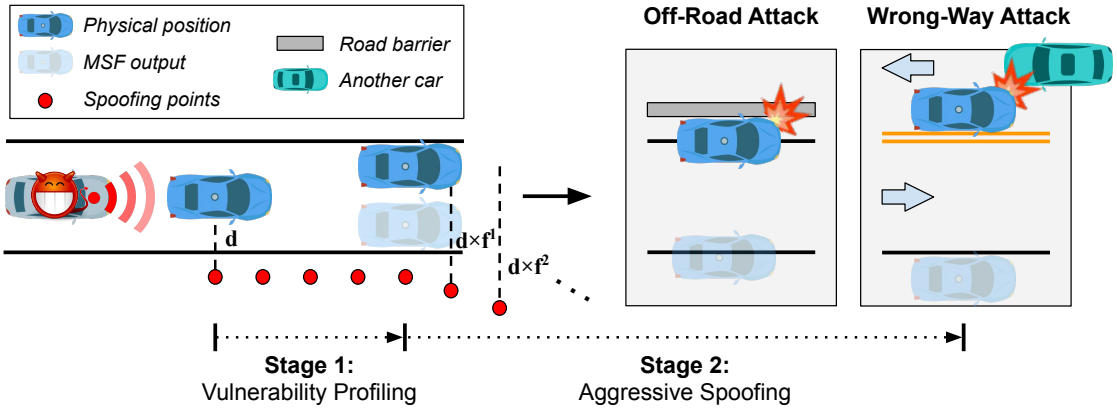

Figure 3.1: Illustration of the 2-stage attack design and consequences of FusionRipper.

Table 3.1: Required deviations for the two attack goals considered in this work. The values are calculated based on common AD vehicle, lane, and road shoulder widths (detailed in Appendix A).

Attack Goal	Required Deviation (m)	
	Local	Highway
Off-Road Attack	0.895	1.945
Wrong-Way Attack	2.405	2.855

sensor traces, and find that it can achieve high effectiveness (over 97% and 91.3% success rates) for both off-road and wrong-way attacks. We also find that such high effectiveness is robust to various practical factors.

- To improve the attack practicality, we further design an offline attack parameter profiling method that can effectively identify attack parameters with 84.2% and 80.7% success rates for off-road and wrong-way attacks respectively, with the profiling cost of at most half a day. We also discuss promising defenses directions.

3.2 Attack Model and Problem Formulation

3.2.1 Attack Goal and Incentives

Attack goals. In this work, we target an attack scenario where an attack vehicle tailgates a victim AD vehicle while launching a GPS spoofing attack, which is both practical and effective as evaluated by previous work using real cars [414]. In such a scenario, we consider an attack goal of introducing large *lateral* deviations to the localization output of the victim AD vehicle, i.e., deviating to the left or right. Since all vehicles need to drive within their designated road lanes for safety protections, such lateral deviations can pose a direct threat to road safety.

In particular, we consider two concrete attack goals specific to the AD context: *off-road attacks* and *wrong-way attack*. As illustrated in Fig. 3.1, the former aims at deviating to either left or right until the victim drives off the road pavement, while the latter aims at deviating to the left until the victim drives on the opposite traffic lane. Table 3.1 lists the required deviations to achieve these two goals, which will be used in our subsequent security analysis.

In the AD context, these two attack goals can cause various *safety hazards specific to localization errors* such as driving off road to hit road curbs or falling down the highway cliff. Since in high-level AD systems the perception module is only designed for obstacle detection and the localization module is in full charge of identifying road deviations [28, 29, 31, 7, 213], these hazards cannot be prevented even when the perception module is functioning perfectly. Moreover, such hazards cannot be prevented even if high-level AD systems directly use perception sensors, e.g., cameras and ultrasonic sensors, for collision avoidance. These two attack goals can also cause *vehicle collisions*, e.g., with vehicles in adjacent or opposite traffic lanes. Even when the AD vehicle can perform automatic emergency brake, it cannot

avoid being hit by other vehicles that fail to yield on time, especially those human driving ones with over 2 sec average driver reaction time [12].

Attack incentives. No matter whether road accidents are caused, the victim AD vehicles under the two attack goals are already violating the traffic rules [14, 13] and exhibiting unsafe driving behaviors. These can already damage the reputation of the corresponding AD company. Thus, a likely attack incentive is *business competition*, which can allow one AD company to deliberately damage the reputation of its rival AD companies and thus unfairly gain competitive advantages. This is especially realistic today considering that there are over 40 companies competing in the AD market [3]. Meanwhile, considering the direct safety impact, we also cannot rule out the possible incentives for terrorist attacks or targeted murders, e.g., against civilians, or controversial politicians or celebrities.

3.2.2 Threat Model

Attacker's capability. We assume that the attacker can launch GPS spoofing (§2.1.2) to control the positioning measurements of the victim's GPS receiver, with a similar level of measurement uncertainty as the natural GPS signals. We also assume that the attacker can track the physical positions of the victim AD vehicle in the real time during the tailgating. This can be achieved by computing the attack vehicle's own position and offsetting it with the relative position between the attack vehicle and the victim. One concrete scenario is that the attack vehicle is also an AD vehicle with a similar set of sensors and run state-of-the-art AD localization algorithms for its own position and AD perception algorithms for the relative position. Under this scenario, the attacker can thus accurately track the victim positions since for AD vehicles precisely tracking the positions of surrounding obstacles in the real time is one of the most basic tasks for ensuring correct and safe driving. Such a scenario is especially realistic when the attack is from rival AD companies (incentive discussed in §3.2.1).

AD control assumption. We assume that AD systems are designed to drive on the center of traffic lanes, and constantly tries to correct any deviation to the center. State-of-the-art AD systems from both the academia [295] and industry [7, 213] follow such design and use lateral controllers to enforce it at a high frequency in the control module (e.g., 100 Hz in Apollo [7]). This means that when the attacker introduces a deviation to the MSF output (e.g., to the right in Fig. 3.1), the victim AD vehicle will actively correct it and thus cause its physical-world position to have the same amount of deviation but to the *opposite* direction (e.g., to the left in Fig. 3.1).

3.2.3 Attack Formulation

Based on the attack model above, the attack in our study can be formulated as the following optimization problem:

$$\begin{aligned} \max_{\{\delta_k^a | k=1,...,n\}} \quad & \mathcal{D}(x_n^a, \{x_k | k = 1, ..., n\}) \\ \text{where} \quad & x_k^a = \mathcal{M}(x_{k-1}^a, r_k + \delta_k^a, z_k^{\text{lidar}}, imu_k), x_0^a = x_0, \end{aligned} \tag{3.1}$$

where δ_k^a is the GPS spoofing distance to the victim's physical-world position r_k on the road plane, x_k is the MSF output without the attack, x_k^a is the MSF output with the attack, z_k^{lidar} is the LiDAR locator output, imu_k is the IMU measurement, $\mathcal{D}(\cdot)$ denotes the lateral deviation between a position and a trajectory, and $\mathcal{M}(\cdot)$ denotes an iteration in the KF-based MSF algorithm (introduced in §2.1.1), and k is the iteration index. As shown, mathematically our attack on MSF is to find a sequence of spoofing distances $\{\delta_k^a | k = 1, ..., n\}$ that can maximize the deviation of the n-th attacked MSF output to the original trajectory $\{x_k | k = 1, ..., n\}$.

3.3 Security Analysis of MSF Algorithm

To systematically understand the security property of MSF-based AD localization, we start with the necessary first step: understanding the upper-bound attack effectiveness, i.e., the maximum possible deviation, under the attack formulation.

3.3.1 Upper-Bound Attack Effectiveness

Analysis methodology. To analyze the upper-bound attack effectiveness, we perform exhaustive search of possible attack inputs $\{\delta_k^a | k = 1, ..., n\}$ to the representative MSF implementation, BA-MSF, to find the one that can maximize Eq. 3.1. We did not choose to use an optimizer since the BA-MSF implementation is released in the binary form and thus we cannot directly get its analytical formula. For a given sensor input trace in our analysis, there are multiple possible *attack windows*, i.e., from one GPS input to another later. For each attack window, we iteratively search for the δ_k^a that can deviate the most from x_k, which is a method also used in previous theoretical work on the security of single-source KF [345, 249, 274, 273]. In accordance with our threat model, we set the measurement uncertainty of GPS spoofing inputs as the median value in real-world sensor input traces of BA-MSF.

We perform the analysis above on two types of sensor input traces: (1) real-world trace, and (2) synthetic noise-free trace. The former is obtained by directly recording the run-time MSF input while the AD vehicle is driving in the real world. Analysis results from this type of traces have the highest realism, but the types of analysis we can perform are limited since we cannot easily modify the sensor data without violating the consistency among different sensor inputs, and the analysis insights can be less clean due to real-world sensor noises. Thus, we complement it with the latter, which synthesizes MSF inputs following

a given driving trajectory, with all the LiDAR locator and non-spoofed GPS inputs set to the ground truth positions, their measurement uncertainty set to the medium value in the real-world trace, and the IMU measurements calculated according to the driving trajectory.

Experimental setup. We obtain the official BA-MSF implementation from the Apollo AD system code base [7]. For the real-world trace, we use the BA-MSF input trace released by Apollo, which is recorded in Sunnyvale, CA and 4-min long [4]. In this work, we denote it as *ba-local*. For the synthetic trace, we generate one for a common driving trajectory: driving on a straight road with a constant velocity of 45 mph. In our analysis, we use an attack window of 10 attack inputs, which is 10 seconds since the GPS input is 1 Hz in Apollo. In the exhaustive search, we enumerate δ_k^a from 0 to 10 meters with step size of 0.04 meters on both left and right sides, since we find that in our experiments GPS input deviations larger than that are identified as outliers by the Chi-squared test in BA-MSF. The medium measurement uncertainty values for GPS and LiDAR locator are calculated from trace *ba-local*.

Results. Fig. 3.2 (a) shows the distribution of the upper-bound deviations achieved in the 10-point attack windows for each trace. As shown, in both real-world and synthetic traces, even such maximum possible attack effectiveness is very limited: majority (76.0%) of the attack windows in the real-world trace and *all* of those in the synthetic trace cannot reach even the lowest required deviations (0.895 m) in Table 3.1. The main reason behind such poor attack performances is as follows. First, due to outlier detection, the maximum deviation achievable by the first attack input is very small, e.g., at most 0.06 meters. Next, such tiny deviation can be quickly corrected by LiDAR locator inputs since in between two GPS attack inputs there are 5 LiDAR locator inputs (5 Hz in Apollo). This makes it highly difficult for subsequent attack inputs to build upon the deviations achieved by previous attack input. Thus, production-grade KF-based MSF algorithms today can indeed generally enhance the security against GPS spoofing.

At the same time, we also observe that the results between the real-world trace and the

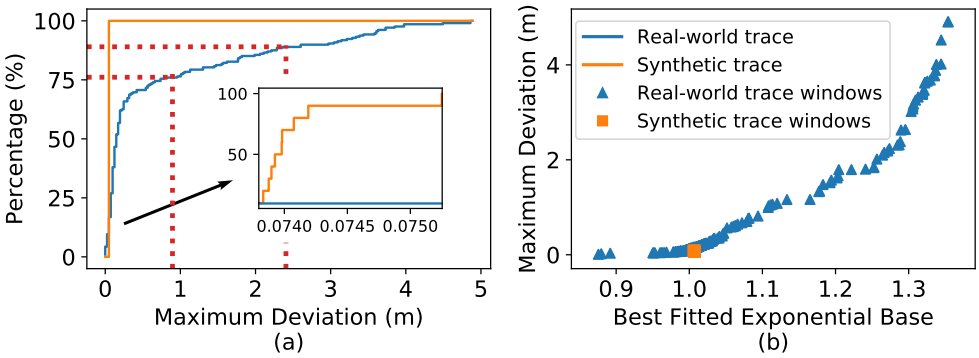

Figure 3.2: (a) CDF of the maximum deviations for attack windows in real-world and synthetic traces. Attack goals are marked in red dotted lines. (b) Maximum deviations and best fitted exponential bases of attack windows in the two traces.

synthetic trace have very sharp differences: in the synthetic trace, the upper-bound deviations for all attack windows are at most 0.076 meters, while those in the real-world trace are generally larger, with 90.3% of them larger than 0.076 meters. This suggests that *sensor noises in the real world can generally degrade the security of MSF*. As shown later, such real-world factors can actually enable highly effective attacks that fundamentally break MSF in practice.

Observation: take-over effect. While our results show a general lack of attack capability to achieve even the easiest attack goal in Table 3.1, we also observe that for the real-world trace there still exist 14% attack windows that can actually achieve over 2 meters deviations, which are large enough for some of our attack goals. For all of these windows, we find that GPS spoofing is able to cause *an exponential growth* of deviations, and one such example is shown on the left of Fig. 3.3. As shown, its deviation trend is very different from those in majority of other attack windows as shown on the right of Fig. 3.3, which is almost flat.

To more quantitatively measure such observation, for each window, we fit an exponential function $f(x) = a^x + b$ to the deviations, where x is the x-th attack point and $f(x)$ is the deviations. For each 10-point window, we use the exponential base a in the best fitted function (based on the mean squared error) to measure the exponential growth trend. As

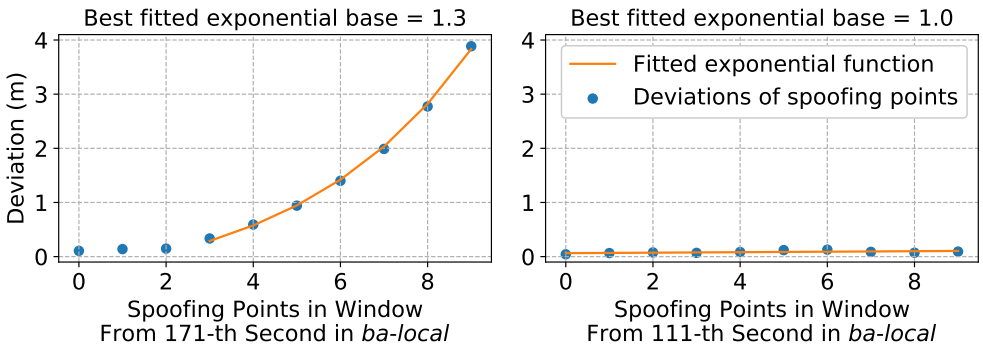

Figure 3.3: The deviations and best fitted exponential bases of two example attack windows in the real-world trace. Left is with take-over effect; Right is without take-over effect.

shown in Fig. 3.2 (b), such exponential growth trends have strict positive correlation with the upper-bound deviations in the attack windows, and all windows that can have very large deviations, e.g., over 3 meters for achieving *all* attack goals in Table 3.1, have very clear exponential growth trend, e.g., with a being at least 1.3 (the trend on the left of Fig. 3.3).

Such exponential growth trend is very similar to the situation when the spoofed GPS is the only positioning source in KF updates, which is confirmed by re-running the upper-bound attack analysis in the synthetic trace without LiDAR locator inputs as shown in Fig. 3.4. This means that for these windows with exponential deviation growths, GPS inputs somehow become the dominating KF update source (we will analyze the cause later). In fact, according to the Chi-squared test values in the analysis logs, we find that LiDAR locator inputs actually become outliers in the latter parts of these windows and then can not provide corrections anymore. This thus *fundamentally defeats the design principle of MSF, i.e., the fusion of multiple input sources for more robustness and accuracy.* In this work, we call it *take-over effect.*

For an attacker, such take-over effect is the most desired attack outcome, since it can efficiently cause *arbitrary deviations* and thus lead to both off-road and wrong-way attacks, and even larger ones if desired. Thus, in the next section we perform a cause analysis to understand why such take-over effect appears in the real-world trace.

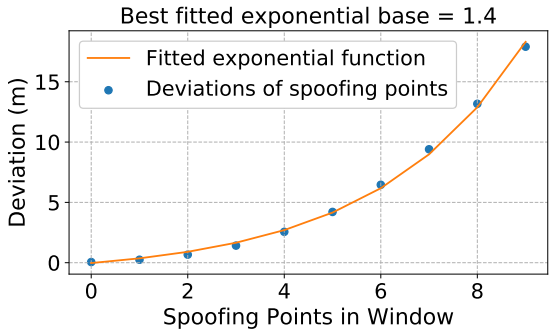

Figure 3.4: The deviation growth and the best fitted exponential base for BA-MSF with only the spoofed GPS input in KF updates (or a *single-source* KF-based MSF) in the synthetic trace under exhaustive search.

3.3.2 Cause Analysis

Since take-over effect does not appear in all attack windows, there must be some factors other than the attack input δ_k^a that contribute to the take-over opportunity. To analyze the causes for take-over effect, we first identify possible contributing factors using theoretical analysis and experimental validation, and then use correlation analysis to identify the most important factors for the observed take-over effect in our analysis.

Derivation of contributing factors. To identify the possible contributing factors to the deviations in MSF, we first perform theoretical analysis based on the general KF-based MSF design (§2.1.1). For the ease of the theoretical analysis without loss of generality, we target the smallest unit in the attack, the MSF operation pipeline between two consecutive GPS spoofing inputs, and simplify it to only have one IMU input and one LiDAR locator input. Fig. 3.5 shows such simplified pipeline, and the notations we use in the analysis, where dev_1, dev_{imu}, dev_{lidar}, and dev_2 denote the MSF output deviations after the first GPS spoofing input, the IMU input, the LiDAR locator input, and the second GPS spoofing inputs.

Here we derive the deviations after each step in the simplified but general KF-based MSF operation pipeline for theoretical contributing factor analysis. The math notations used in the derivation are listed in Table 3.2.

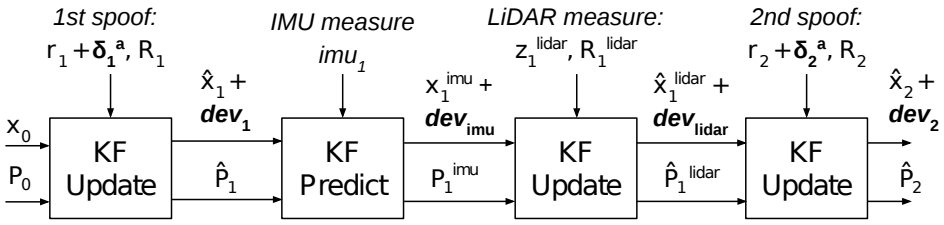

Figure 3.5: A simplified but general MSF operation pipeline under GPS spoofing attack for theoretical analysis.

Table 3.2: Notations in KF and contributing factor derivation.

Notation	Description
$\mathbf{x}$	KF state, e.g., PVA
$\mathbf{P}$	KF state co-variance, i.e., state uncertainty
$\mathbf{K}$	Kalman gain of the measurement
$\mathbf{F}$	State transition model; it describes the kinematics functions used in KF prediction
$\mathbf{H}$	Observation model; it is an identity matrix if the measurement and state have the same scale
$\mathbf{Q}$	Process noise co-variance; usually a fixed pre-tuned matrix
$\mathbf{z}$	Sensor measurement
$\mathbf{R}$	Measurement variance, i.e., measurement uncertainty
$\mathbf{r}$	Victim's physical position
δ	Spoofing distance to victim's physical position
Δ	LiDAR position distance to the original MSF position
dev	The deviation after each KF operation under spoofing

First, after spoofing the 1st GPS point with a spoofing distance δ_1^a, the KF state equations become:

$$\begin{aligned}
\hat{\mathbf{x}}_1^a &= \mathbf{x}_0 + \mathbf{K}_1(\mathbf{r}_1 + \delta_1^a - \mathbf{H}\mathbf{x}_0) \\
&= \hat{\mathbf{x}}_1 + \mathbf{K}_1\delta_1^a \\
\hat{\mathbf{P}}_1 &= \mathbf{P}_0 - \mathbf{K}_1\mathbf{H}\mathbf{P}_0 \\
\mathbf{K}_1 &= \mathbf{P}_0\mathbf{H}^T(\mathbf{H}\mathbf{P}_0\mathbf{H}^T + \mathbf{R}_1)^{-1}
\end{aligned}$$

Thus, the deviation after spoofing the first point is:

$$dev_1 = \mathbf{K}_1\delta_1^a$$

Second, we perform an IMU prediction. IMU values are used in the kinematics function described by the matrix $\mathbf{F}_1$:

$$\begin{aligned}
\mathbf{x}_1^{\text{imu},a} &= \mathbf{F}_1\hat{\mathbf{x}}_1^a \\
&= \mathbf{F}_1(\hat{\mathbf{x}}_1 + \mathbf{K}_1\delta_1^a) \\
&= \mathbf{x}_1^{\text{imu}} + \mathbf{F}_1\mathbf{K}_1\delta_1^a \\
\mathbf{P}_1^{\text{imu}} &= \mathbf{F}_1\hat{\mathbf{P}}_1\mathbf{F}_1^T + \mathbf{Q}
\end{aligned}$$

After the IMU prediction, the deviation becomes:

$$dev_{\text{imu}} = \mathbf{F}_1 dev_1$$

Third, a LiDAR locator update is applied. $\Delta_{\text{lidar}} = \mathbf{x}_1^{\text{imu}} - \mathbf{z}_1^{\text{lidar}}$ describes the distance between the LiDAR position and the original non-spoofed KF state. This is because sensor noises or LiDAR locator inaccuracies will cause LiDAR locator outputs to be misaligned

with the MSF output.

$$
\begin{aligned}
\hat{\mathbf{x}}_1^{\text{lidar},a} &= \mathbf{x}_1^{\text{imu},a} + \mathbf{K}_1^{\text{lidar}}(\mathbf{z}_1^{\text{lidar}} - \mathbf{H}\mathbf{x}_1^{\text{imu},a}) \\
&= \mathbf{x}_1^{\text{imu}} + dev_{\text{imu}} \\
&\quad + \mathbf{K}_1^{\text{lidar}}(\mathbf{z}_1^{\text{lidar}} - \mathbf{H}(\mathbf{x}_1^{\text{imu}} + dev_{\text{imu}})) \\
&= \hat{\mathbf{x}}_1^{\text{lidar}} + dev_{\text{imu}} - \mathbf{K}_1^{\text{lidar}}(\Delta_{\text{lidar}} + dev_{\text{imu}}) \\
\hat{\mathbf{P}}_1^{\text{lidar}} &= \mathbf{P}_1^{\text{imu}} - \mathbf{K}_1^{\text{lidar}}\mathbf{H}\mathbf{P}_1^{\text{imu}} \\
\mathbf{K}_1^{\text{lidar}} &= \mathbf{P}_1^{\text{imu}}\mathbf{H}^T(\mathbf{H}\mathbf{P}_1^{\text{imu}}\mathbf{H}^T + \mathbf{R}_1^{\text{lidar}})^{-1}
\end{aligned}
$$

LiDAR locator output provides correction on the deviation. After the KF update, the deviation then becomes:

$$
dev_{\text{lidar}} = dev_{\text{imu}} - \mathbf{K}_1^{\text{lidar}}(\Delta_{\text{lidar}} + dev_{\text{imu}})
$$

Finally, we spoof the second GPS point with the spoofing distance δ_2^a:

$$
\begin{aligned}
\hat{\mathbf{x}}_2^a &= \hat{\mathbf{x}}_1^{\text{lidar},a} + \mathbf{K}_2(\mathbf{r}_2 + \delta_2^a - \mathbf{H}\hat{\mathbf{x}}_1^{\text{lidar},a}) \\
&= \hat{\mathbf{x}}_1^{\text{lidar}} + dev_{\text{lidar}} \\
&\quad + \mathbf{K}_2(\mathbf{r}_2 + \delta_2^a - \mathbf{H}(\hat{\mathbf{x}}_1^{\text{lidar}} + dev_{\text{lidar}})) \\
&= \hat{\mathbf{x}}_2 + dev_{\text{lidar}} + \mathbf{K}_2(\delta_2^a - dev_{\text{lidar}}) \\
\hat{\mathbf{P}}_2 &= \hat{\mathbf{P}}_1^{\text{lidar}} - \mathbf{K}_2\mathbf{H}\hat{\mathbf{P}}_1^{\text{lidar}} \\
\mathbf{K}_2 &= \hat{\mathbf{P}}_1^{\text{lidar}}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_1^{\text{lidar}}\mathbf{H}^T + \mathbf{R}_2)^{-1}
\end{aligned}
$$

And the deviation after the second spoofing point will be:

$$
dev_2 = dev_{\text{lidar}} + \mathbf{K}_2(\delta_2^a - dev_{\text{lidar}})
$$

Based on the derivation, there are 4 theoretical contributing factors to dev_2 besides the attack input δ_k^a:

- *Initial MSF state uncertainty* ($\mathbf{P}_0$): The larger $\mathbf{P}_0$ is, the less confident the MSF algorithm has on its positioning output, and thus more updates are taken from attack inputs δ_k^a, leading to larger dev_2.

- *LiDAR measurement uncertainty* ($\mathbf{R}_1^{\text{lidar}}$): The larger R_1^{lidar} is, the less confident the LiDAR locator is on its positioning output z_1^{lidar}, and thus the larger the *remaining* deviation after LiDAR locator's correction dev_{lidar}, leading to larger dev_2.

- *Difference between LiDAR position and the original MSF output without attack* (Δ_{lidar}): The impact of Δ_{lidar} on dev_2 has two phases. First, as Δ_{lidar} increases, the correction from the LiDAR update increases, which causes dev_{lidar} being smaller and decreases dev_2. Second, after Δ_{lidar} becomes too big that makes z_1^{lidar} an outlier, no correction can be applied any more and thus dev_2 becomes larger than before. Thus, there is a non-linear relationship between Δ_{lidar} and dev_2.

- *IMU measurement* (imu_1): imu_1 affects on dev_2 in two ways. First, imu_1 is used in $\mathbf{F}_1$ (the IMU-based integration function in Eq. 2.1), which directly affects dev_{imu} and further affects dev_2. Second, $\mathbf{F}_1$ affects $\mathbf{P}_{\text{imu}}$ and then the Kalman gain at LiDAR update $\mathbf{K}_{\text{lidar}}$ and at the second spoofing $\mathbf{K}_2$ (Eq. 2.2). Note that larger $\mathbf{K}_{\text{lidar}}$ means larger correction and thus smaller dev_2, while larger $\mathbf{K}_2$ means larger dev_2. Thus, the relationship between imu_1 and dev_2 depends on the design of $\mathbf{F}_1$ and the competition of the impact of larger $\mathbf{K}_{\text{lidar}}$ and larger $\mathbf{K}_2$ on dev_2.

Experimental validation of derived contributing factors. To validate whether these 4 factors indeed affect the actual BA-MSF implementation, we take a segment from the synthetic sensor trace as shown in Fig. 3.6, and modify different parts of sensor data to

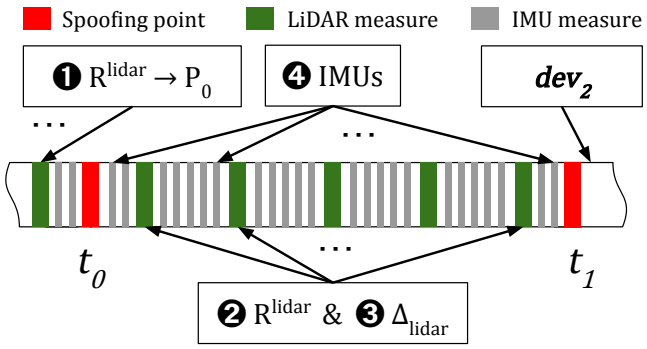

Figure 3.6: Modeling of each factor in the synthetic trace. We modify different parts of the sensor data in order to observe how the factors affect the 2nd deviation dev_2.

model the change of the four contributing factors. As shown, the segment consists of two GPS spoofing points. Since no spoofing has been applied prior to time t_0, the deviation prior to t_0 is zero. Unlike the simplified MSF operation pipeline considered in the theoretical derivation, we apply the original KF prediction and update sequences as real-world sensor traces for BA-MSF, i.e., 1 Hz for GPS, 5 Hz for LiDAR locator, and 200 Hz for IMU [7].

For each contributing factor, we measure the deviation after the 2nd spoofing point to understand the relationship with the deviation. To eliminate the influence from the GPS spoofing distance, we exhaustively search for different distances for two GPS spoofing points and use the best one in our results. We use the median value of the GPS uncertainty in *ba-local* as the uncertainty values for the GPS spoofing points, which is the same as in §3.2.2.

Validation results. The experiment results are shown in Fig. 3.7 and described below:

- For $\mathbf{P}_0$, there is no direct way to modify it since it is not part of sensor data. Here we vary $\mathbf{R}^{\text{lidar}}$ before t_0 to *indirectly* generate different values of $\mathbf{P}_0$. Since the LiDAR locator outputs are aligned with the MSF state (i.e., $\Delta_{\text{lidar}} = 0$) before t_0, the change of $\mathbf{R}^{\text{lidar}}$ will only affect $\mathbf{P}_0$. As shown in the top-left subfigure in Fig. 3.7, the results validate that a larger $\mathbf{P}_0$ will cause a larger deviation dev_2.

- We modify $\mathbf{R}^{\text{lidar}}$ between time t_0 and t_1 to observe how it affects the correction capa-

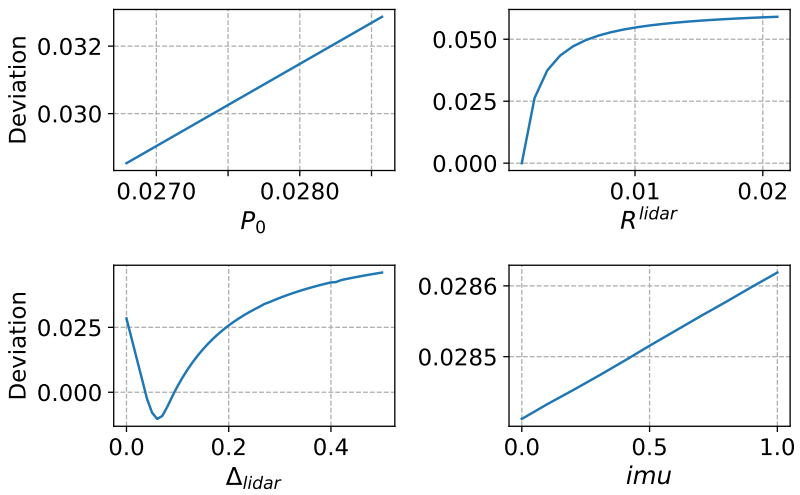

Figure 3.7: Relationship between the contributing factors and the spoofing deviation in the synthetic trace.

bility of LiDAR locator outputs on the deviation. As shown in the top-right subfigure in Fig. 3.7, our results validate that $\mathbf{R}^{\text{lidar}}$ has a positive effect on the deviation, but reaches a plateau when it is overly large.

- The modeling of Δ_{lidar} is straightforward: We directly set the LiDAR locator outputs to have different distances to the MSF state. The bottom-left subfigure in Fig. 3.7 shows our results. As shown, aligned with our theoretical analysis, a small Δ_{lidar} can correct the deviation introduced by the first GPS spoofing. However, when Δ_{lidar} increases, at a certain point it causes the MSF output to deviate to the opposite direction. This is because Δ_{lidar} provides a large update to the velocity component in the MSF state such that the deviation is over-corrected as time accumulates. When Δ_{lidar} becomes even larger, the deviation starts to increase since LiDAR locator outputs become outliers, which also conforms to our theoretical analysis.

- For *imu*, we modify the acceleration component in the IMU measurements between time t_0 and t_1. In addition, we align the LiDAR locator positions to the *non-spoofed* MSF outputs during this period to ensure that $\Delta_{\text{lidar}} = 0$. As shown in the bottom-right subfigure in Fig. 3.7, our results show that *imu* has a positive influence on the

deviation overall, which shows that the impact of $\mathbf{K}_2$ is much larger than the impact of $\mathbf{K}_{\text{lidar}}$ to dev_2.

Factor importance analysis. With the 4 contributing factors identified, we then use popular causality analysis methods to understand the importance of these factors on causing the take-over effect observed in §3.3.1. Specifically, we perform the exponentiation function fitting as described in §3.3.1, and label the windows with exponential base a over 1.1 as windows with take-over effect. As shown in Fig. 3.2 (b), for windows without any take-over effect, e.g., the ones for the synthetic trace, the exponential base a is way below 1.1. With the exponential fitting results, we identify the first point of the exponential growth to obtain $\mathbf{P}_0$. For $\mathbf{R}^{\text{lidar}}$, Δ_{lidar}, and imu, we use the average values from the first point of the exponential growth to the end of the window. We use 2 statistical testing methods commonly used for causality analysis [269, 296, 107]: Pearson's Correlation and Fisher's Exact Test.

Analysis results. Table 3.3 shows the experiment results. For the two statistical testing methods, $p < 0.05$ is considered statistically significant, and $r > 0.5$ and $or > 9$ are considered strongly correlated for Pearson's Correlation and Fisher's Exact Test respectively [128]. As shown, only the p values for $\mathbf{P}_0$ and $\mathbf{R}^{\text{lidar}}$ are statistically significant for both methods, with their r values very close to showing strong correlations, and their or values showing strong correlations. In contrast, neither of the r or or values for Δ_{lidar} and imu show strong correlations, and for imu, the results are not even statistically significant. This suggests that the take-over effect we observe in our upper-bound analysis is most likely caused by relatively large $\mathbf{P}_0$ and $\mathbf{R}^{\text{lidar}}$ in the corresponding attack windows.

For these two most important contributing factors, $\mathbf{R}^{\text{lidar}}$ reflects the lack of confidence in the LiDAR-based localization algorithm during the attack window, and $\mathbf{P}_0$ reflects the lack of confidence in the KF states at the beginning of the attack window. This means that *take-over opportunities, or vulnerabilities, appear when the MSF is in relatively unconfident*

Table 3.3: Correlations between the contributing factors and the take-over vulnerability. Results with statistically strong correlation are highlighted in bold.

Correlation Method	Factor Importance			
	$\mathbf{P}_0$	$\mathbf{R}^{\text{lidar}}$	Δ_{lidar}	*imu*
Pearson's Correlation	**0.42 (2.0e-10)**	**0.44 (3.5e-11)**	0.12 (8.4e-2)	0.01 (8.6e-1)
Fisher's Exact Test	**21.09 (8.6e-6)**	**11.78 (5.2e-8)**	5.91 (3.2e-4)	1.95 (1.1e-1)

Pearson's correlation: r (p-value), where r is the correlation coefficient
Fisher's exact test: *or* (p-value), where *or* is the odds ratio

periods. Because of this, the MSF algorithm needs to take more updates from the GPS inputs, the relatively most confident input source in that period, which thus allows GPS inputs to dominate KF updates and trigger the take-over effect.

Since $\mathbf{R}^{\text{lidar}}$ is the uncertainty reported by LiDAR locator, a large $\mathbf{R}^{\text{lidar}}$ is caused by the inaccuracies of such locator algorithm in practice. From the KF equations (§2.1.1), a large $\mathbf{P}_0$ is mainly caused by larger uncertainties from the LiDAR locator and GPS updates before the attack window, which is thus due to algorithm inaccuracies in LiDAR locator and noises in GPS signals. Thus, unconfident periods in MSF are mainly created by practical factors such as algorithm inaccuracies and sensor noises. This also explains why we cannot observe any take-over effect in synthetic noise-free trace. These practical factors are fundamentally difficult to avoid in practice, which is exactly why MSF is designed to compensate such inaccuracies and noises from individual sources [28, 386, 170, 346, 357, 331, 140, 340, 233, 214]. However, as shown in our analysis, *even for the high-end sensors used in AD vehicles today, these inaccuracies and noises are unfortunately large and frequent enough for GPS spoofing to exploit and fundamentally break MSF in practice.*

3.4 Attack Design: FusionRipper

Although our analysis in §3.3 reveals that there do exist take-over vulnerabilities for MSF in the real world, such vulnerabilities only appear in the unconfident periods created by dynamic and non-deterministic practical factors such as algorithm inaccuracies and sensor noises, which is not observable by the attacker in a tailgating attack vehicle (§3.2.2) and are highly difficult, if not impossible, to directly control. Thus, the attacker has to *opportunistically* capture and exploit such vulnerable periods in the actual attack time.

Leveraging this idea, we propose a novel attack design against MSF-based AD localization, called *FusionRipper*, which consists of 2 stages as depicted in Fig. 3.1:

Stage 1: Vulnerability profiling. In this stage, the attacker performs GPS spoofing and measures the feedback from the victim AD vehicle to profile when vulnerable periods appear. In our design, we aim for as fewer attack parameters as possible to maximize the ease of implementation and robustness, and thus choose to use *constant spoofing* for this stage, i.e., always setting δ_k^a to a constant d as shown in Fig. 3.1. Although such profiling method is simple, our evaluation results later in §3.5 show that it is able to achieve a high attack success rate that is very close to the theoretical upper bound.

While performing constant spoofing, the attacker tracks victim's physical positions in real time and measures their deviations to the center of traffic lane (described in §3.2.2). If such deviation is as large as causing the AD vehicle to exhibit unsafe driving behaviors, e.g., about to have unnecessary lane straddling, the victim AD vehicle is considered as in the vulnerable period. Our design uses the deviation that can touch the left or right lane line on local roads (0.295 meters, detailed in Appendix A) as the threshold to determine vulnerable periods. The intuition is that a properly designed and tested AD system should very rarely have large position deviations that can cause unsafe driving behaviors under normal fluctuations of sensor inputs. For example, the errors of BA-MSF evaluated by Baidu Apollo AD vehicles

on real roads are within 0.054 meters [386], which is far less than 0.295 meters. Thus, when such rare deviation appears, it is very likely caused by the constant spoofing, and the MSF algorithm is very likely in an unconfident period since it takes larger update from the spoofed GPS inputs.

Stage 2: Aggressive spoofing. After the vulnerable period is identified, the attacker can then perform aggressive spoofing to trigger the take-over effect and thus quickly induce large deviations. As shown in our security analysis in §3.3.1, the deviations grow exponentially during the take-over effect, and thus we choose exponential spoofing in the aggressive spoofing stage. As shown in Fig. 3.1, as soon as the attacker identifies a vulnerable period, she switches to use spoofing distance $d \times f^i$, where an exponential base f is cumulatively multiplied to previous spoofing distance at each of the spoofing points, and i is the index of the aggressive spoofing inputs.

Generality. Since FusionRipper is designed to exploit the take-over vulnerability that is general to any KF-based MSF as discussed in our cause analysis based on the general form of KF-based MSF (§3.3.2), its design is generally applicable to any KF-based MSF algorithms. As shown in our generality evaluation later (§3.5.4), FusionRipper is highly effective on different KF-based MSF designs and implementations.

3.5 Attack Evaluation

3.5.1 Evaluation Methodology

Experimental setup. Following the common practice among AD companies [167, 169], we evaluate FusionRipper on real-world sensor traces. Specifically, we use the real-world trace *ba-local* used in our security analysis earlier (§3.3), and also traces from KAIST Complex

Urban [203], a dataset for evaluating AD systems. Since *ba-local* is collected by the Apollo team and is designed specifically for evaluating MSF-based localization algorithms for Apollo, it is by default compatible with BA-MSF with a complete positioning sensor set as well as the HD Map for running the LiDAR locator[1].

Similar to *ba-local*, the traces in the KAIST dataset are also collected by high-end AD-grade positioning sensors [203]. But unfortunately, they do not provide the HD Map for running the LiDAR locator in BA-MSF. To address this, we assume an *ideal* LiDAR locator which always outputs the ground truth positions provided in the KAIST dataset, with their measurement uncertainty set to the median value of that in *ba-local*. Considering that one of the likely causes for the take-over effect is the LiDAR locator inaccuracies, especially the measurement uncertainty values (§3.3.2), this assumption only makes the attack harder and thus the results will provide the worst-case attack effectiveness on the KAIST traces.

Trace selection in KAIST dataset. The KAIST dataset includes 18 local traces and 2 highway traces that are compatible with BA-MSF, and we select 3 local ones and both the 2 highway ones. We truncate them to the first 5 minutes to keep the evaluation time manageable. In the selection of local traces, we select the ones with the smallest average MSF state uncertainty (i.e., most confident). Table 3.4 shows the average MSF state co-variance value, i.e., uncertainty, when running BA-MSF on the 20 traces in the KAIST dataset that (1) have the complete sensor data needed by BA-MSF, e.g., some KAIST traces do not have complete IMU data, and (2) from a stationary position to provide a complete motion history, which is required for BA-MSF to have stable outputs. Among the 18 local traces and 2 highway traces, we choose both the eligible highway traces, and select the top 3 from the local traces with the lowest MSF state uncertainties. Considering that state uncertainty is one of the two most important contributing factors to the take-over effect (§3.3.1), the evaluation results on these traces will provide the worst-case attack effectiveness for the

[1]Apollo released 8 sensor traces recorded with localization, but only *ba-local* has both the complete sensor set and compatible format with BA-MSF.

Table 3.4: Average MSF co-variance, i.e., uncertainty, of the KAIST local and highway traces. We ranked the traces based on their MSF state co-variance (the lower the more confident), and pick the most confident ones (in **bold**) in our evaluation.

Local Trace	Avg. MSF Co-variance	Rank	Local Trace	Avg. MSF Co-variance	Rank
ka-local08	**0.0032**	**1**	*ka-local39*	0.1143	10
ka-local31	**0.0080**	**2**	*ka-local16*	0.2254	11
ka-local07	**0.0111**	**3**	*ka-local29*	0.3237	12
ka-local37	0.0131	4	*ka-local09*	0.4070	13
ka-local35	0.0146	5	*ka-local14*	0.4468	14
ka-local33	0.0219	6	*ka-local38*	0.8904	15
ka-local36	0.0312	7	*ka-local26*	1.4719	16
ka-local28	0.1026	8	*ka-local27*	6.4191	17
ka-local30	0.1029	9	*ka-local32*	33.3712	18

Highway Trace	Avg. MSF Co-variance	Rank
ka-highway17	**0.0027**	**1**
ka-highway06	**0.0028**	**2**

KAIST traces.

Evaluation metrics. To evaluate the attack effectiveness, we apply attack parameters d and f from all possible attack starting points, i.e., when the GPS input comes, in each trace, since the attacker can discover the victim at any moment in the trace and start performing the attack. As described earlier in §3.4, the attacker switches to aggressive spoofing when the lateral deviation between the spoofed MSF output and the non-spoofed MSF output is over 0.295 meters, which is just about to have lane straddling on local roads.

We consider the attack as successful when the lateral deviation of the MSF output is over the required deviations for the off-road and wrong-way attacks according to Table 3.1. This follows our AD control assumption (§3.2.2), which can directly considers the amount of deviation at the MSF output level as the amount of physical position deviations in the opposite direction to the center line. Later in §3.6.2, we will concretely evaluate this assumption using an end-to-end evaluation with the AD control taking effect. The *success rate* is calculated as the fraction of the successful attack starting points out of all starting points. For each attack starting point, we enumerate the combinations of d from 0.3 to 2.0 meters, with step

Table 3.5: Real-world sensor traces used in our evaluation.

Source	Trace Label	Road Type	Duration	HD Map
Apollo	*ba-local*	Local	257s	Yes
KAIST Complex Urban	*ka-local08*	Local	289s	No
	ka-local31	Local	1014s	
	ka-local07	Local	553s	
	ka-highway17	Highway	1186s	
	ka-highway06	Highway	1937s	

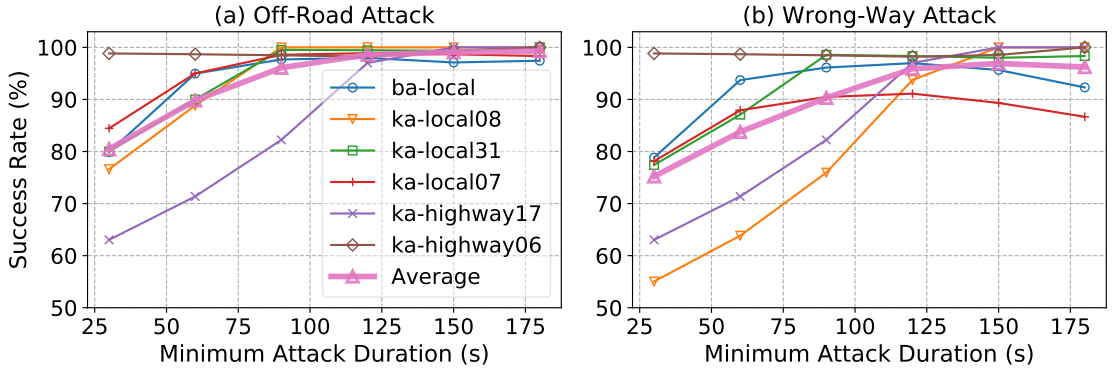

Figure 3.8: Average attack success rates of (a) *off-road* attack and (b) *wrong-way* attack under different minimum attack duration.

size 0.1 meters, and f from 1.1 to 2.0, with step size 0.1. We choose these ranges because we do not find the values out of these ranges can improve the attack effectiveness in our experiments. Each d and f combination is then applied to both the left and right side of the driving direction, since both sides are valid for achieving off-road attack (detailed in §3.2.1). Since it takes time to (1) capture a take-over vulnerability, which is created dynamically and non-deterministically, and (2) reach the required deviations even during take-over effects (§3.3.1), we also consider *minimum attack duration* when calculating success rate, i.e., how much time the attack can last when tailgating the victim AD vehicle. Intuitively, the longer such duration is, the higher chance she can have to hit a vulnerable period.

Table 3.6: Ablation study results on *ba-local* trace.

Attack Configuration	Off-Road		Wrong-Way	
	Succ. Rate	Succ. Time	Succ. Rate	Succ. Time
FusionRipper	98.0%	29s	97.0%	33s
Vulnerability Profiling Stage Only	14.1%	26s	7.0%	29s
Aggressive Spoofing Stage Only	10.1%	8s	5.0%	13s

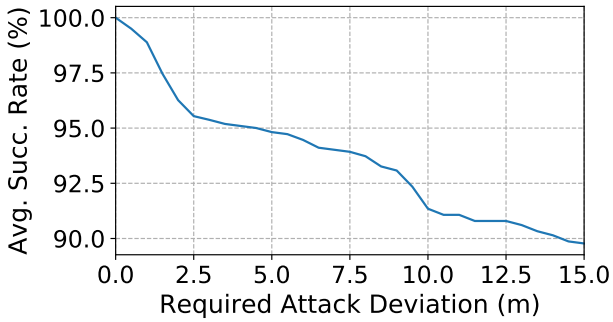

Figure 3.9: Average success rate under different required attack deviations when the minimum attack duration is 2 minutes.

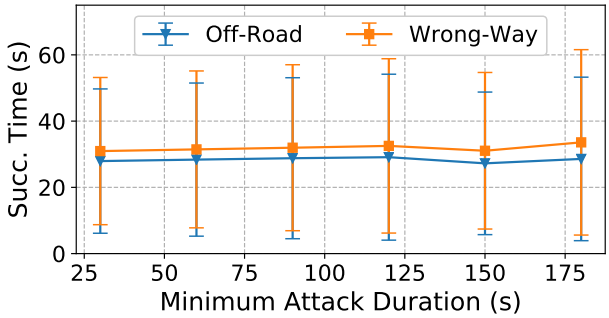

Figure 3.10: Average success time for reaching required deviations in off-road and wrong-way attacks under different minimum attack duration.

3.5.2 Attack Effectiveness

Attack success rates. Fig. 3.8 shows the best success rates of FusionRipper among all the combinations of d and f for the two attack goals. It shows both the results for individual traces and the average result among all traces (the thick pink line). As shown, for all traces, the average success rate is always over 75% for both attack goals even when the minimum attack duration is as low as 30 seconds. When the minimum attack duration increases, the success rates for all traces increase accordingly, which is expected since the attacker has higher chance to capture a vulnerable period. In particular, when the attack can last 2 minutes, *there exists at least one combination of d and f that can achieve over 97% success rate (98.6% on average) for the off-road attack and over 91% success rate (95.9% on average) for the wrong-way attack, for all traces in our evaluation.* Note that this is in fact the worst-case results for KAIST traces as discussed in §3.5.1. Since a normal taxi or truck trip is usually at least 10 minutes, it is highly likely that an attacker can find such a 2-minute tailgating opportunity in practice to launch the FusionRipper attack.

Among all the traces, *ka-local08* and *ka-highway17* shows the lowest success rate in general, especially when the required deviation is large. As shown in Table 3.4, both traces have the smallest average MSF state uncertainty in their categories (i.e., local and highway). This means that their MSF outputs have the highest confidence and thus are the most difficult to attack as we expect in §3.5.1. This also confirms that we are evaluating the worst-case attack effectiveness on KAIST traces.

Between the two attack goals, the success rates only slightly drop for wrong-way attack since it has a larger required deviation. This means that the majority of the captured vulnerable periods have a successful take-over effect that can be exploited to cause different required deviations. To confirm this, we further evaluate the success rates of FusionRipper for even larger required deviations, and find that when the minimum attack duration is 2 minutes,

FusionRipper is able to maintain an average success rate over 91.3% even when the required deviation is 10 meters as shown in Fig. 3.9.

Sensitivity to attack parameters. Table 3.7 lists the top 3 combinations for each trace. As shown, the attack effectiveness of FusionRipper is sensitive to the combinations of d and f. For example, the best d and f combinations are *all different* for the 6 traces. This motivates us to design an offline method to identify effective d and f combinations to increase the attack practicality, which is detailed later in §3.7.

Ablation study. The high attack effectiveness is a result of the combination of the two attack stages. To concretely understand this, we conduct an ablation study on *ba-local*, where we remove one of the two stages in the experiments. For *Vulnerability Profiling Stage Only*, we apply the constant spoofing distance d from each starting point. For *Aggressive Spoofing Stage Only*, we directly scale the spoofing distance using different combinations of d and f from each starting point. For both configurations, we obtain the *highest* success rates by enumerating d or f in the range specified in §3.5.1.

Table 3.6 shows the experiment results for *ba-local* when the minimum attack duration is 2 minutes. As shown, both configurations can only achieve at most 14% and 7% for the two attack goals, which is far less than 98% and 97% by FusionRipper. This means that there are still some very unconfident periods that even stage 1 or stage 2 alone can succeed, but as shown, without the help of each other, the success rate is very limited. This concretely demonstrates the necessity of the current 2-stage design of FusionRipper. Note that FusionRipper has longer attack success time than *Aggressive Spoofing Stage Only* due to the time spent on the vulnerability profiling stage. However, since the current $\sim$30 seconds attack time on average is already quite affordable for a tailgating attacker in practice, such advantage is much less important than the much higher success rates by FusionRipper.

Attack success time. For the attack success time, overall the average success time and

Table 3.7: Top 3 attack parameters with the highest attack success rates when minimum attack duration is 2 min.

Attack	Rank	*ba-local*			*ka-local08*			*ka-local31*			*ka-local07*			*ka-highway17*			*ka-highway06*		
		d	f	Succ. Rate	d	f	Succ. Rate	d	f	Succ. Rate	d	f	Succ. Rate	d	f	Succ. Rate	d	f	Succ. Rate
Off-Road	Top 1	0.6	1.5	98.0%	0.7	1.1	100%	0.5	1.2	99.4%	0.3	1.1	98.9%	0.3	1.2	97.0%	1.1	1.5	98.2%
	Top 2	0.6	1.6	98.0%	0.7	1.2	100%	1.0	1.3	99.4%	0.3	1.2	98.3%	0.3	1.3	97.0%	1.1	1.3	98.2%
	Top 3	0.6	1.7	98.0%	0.7	1.3	100%	1.0	1.4	99.4%	0.4	1.2	98.3%	0.3	1.4	94.0%	1.3	1.3	98.2%
Wrong-Way	Top 1	0.6	1.5	97.0%	0.3	1.2	93.8%	1.0	1.3	98.3%	0.3	1.4	91.1%	0.3	1.2	97.0%	1.2	1.3	98.2%
	Top 2	0.6	1.3	95.0%	0.3	1.3	93.8%	1.0	1.2	97.8%	0.3	1.5	90.6%	0.3	1.3	97.0%	1.3	1.3	98.2%
	Top 3	0.6	1.4	95.0%	0.5	1.3	92.1%	1.1	1.2	97.8%	0.3	1.3	88.3%	0.3	1.4	94.0%	1.1	1.3	97.6%

the standard deviations are very similar under different minimum attack duration as shown in Fig. 3.10. When the minimum attack duration is 2 minutes, the average success time is less than 30 seconds with a standard deviation of around 25 seconds for both off-road and wrong-way attacks. This shows that FusionRipper can generally succeed very fast, e.g., within a minute, even when the attacker plans to attack for over 2 minutes.

3.5.3 Comparison with Naive Attack Method

In this section, we compare FusionRipper with a more naive attack method: *random attack*, which randomly spoofs a deviation within a distance range for each GPS spoofing point.

Experimental setup. We perform experiments by applying FusionRipper and random attack on *ba-local*. In the random attack, we uniformly sample the position deviation between 0 to 10 meters for each spoofing point. The experiments are repeated for 30 trials. In each trial, the spoofing is performed for each attack starting point and on both the left and right. The higher success rate between that of the left and that of the right is taken as the final success rate for each trial.

Results. The first row in Table 3.8 shows the experiment results when the minimum attack duration is 2 minutes. We find that the random attack can barely reach any large deviation, and as shown, its success rates are as low as 3.7% and 0.2% on average for the two attack goals respectively, which are much lower than those from FusionRipper (98.0% and 97%).

3.5.4 Generality of FusionRipper

In this section, we aim at understanding the generality of FusionRipper by evaluating it on more KF-based MSF implementations. Ideally we hope to find other production-grade implementations for AD systems similar to BA-MSF, but to best of our knowledge, BA-MSF

is the only publicly-available one so far. Nevertheless, we still try our best to implement/port and evaluate on two other popular KF-based MSF designs, denoted as *JS-MSF* and *ETH-MSF*, which are both designed for general robotics localization instead of for AD vehicles.

Experimental setup. BA-MSF adopts a Linear KF, the most popular KF design for MSF-based localization (Table 2.1). Thus, we follow a popular Linear KF based MSF design published by Joan Solà [339] and implement JS-MSF. ETH-MSF [154] is an open-source project developed by researchers from ETH Zürich for drones [261], which implements an Extended KF based MSF, the second popular KF design for MSF-based localization (Table 2.1). It has received over 500 stars on GitHub, which is the *highest* among the repositories under the search keyword "kalman filter sensor fusion". Both implementations use a Chi-squared test based outlier detector and directly reject outlier measurements. We follow a common parameter tuning process [177] and reach at most 1.91 and 1.17 meters localization accuracies on *ba-local* for JS-MSF and ETH-MSF respectively. Although such accuracies are far from the centimeter-level accuracy required by AD systems, they are common for general robotics localization [430, 429, 106].

Results. Table 3.8 shows the attack success rates of FusionRipper and random attack on *ba-local* for all 3 KF-based MSF implementations. As shown, FusionRipper can generally achieve high success rates on all three MSFs, which are 100% on both JS-MSF and ETH-MSF for both attack goals. However, we also notice that even random attack can also achieve over 95% success rates for the off-road attack, and over 70% for the wrong-way attack. This suggests that JS-MSF and ETH-MSF are both very unstable, which can also be seen by the fact that their natural localization errors are already 1.17 and 1.91 meters. In contrast, BA-MSF can achieve 0.054 meters accuracy, which is likely due to additional design features such as zero-velocity update [386], and better parameter tuning by professional AD engineers. Thus, while our results show that FusionRipper is general for all 3 KF-based MSF implementations, we believe that the results on BA-MSF can more representatively indicate

Table 3.8: Attack success rates of FusionRipper and random attack on 3 MSF implementations. The attacks are evaluated on *ba-local* with 2-minute minimum attack duration.

Attacked MSF	FusionRipper		Random Attack (avg. of 30 trials)	
	Off-Road	Wrong-Way	Off-Road	Wrong-Way
BA-MSF	98.0%	97.0%	3.7%	0.2%
JS-MSF	100%	100%	97.4%	92.4%
ETH-MSF	100%	100%†	95.9%	72.5%

†Achieves 100% success rate when using a smaller f (1.02).

the security status of production-grade MSF-based AD localization today.

3.6 Practical Attack Considerations

Although FusionRipper already shows very high effectiveness in §3.5, we haven't considered two factors that may affect the attack effectiveness in practice: (1) the variations in the spoofed positions and their measurement uncertainty at the victim's GPS receiver, and (2) sensor input changes due to AD control during the attack. In this section, we evaluate the robustness of FusionRipper under these two practical factors. The experiments in this section are mainly performed on the *ba-local* trace since it has the complete set of real-world sensor inputs for BA-MSF and thus has the highest realism.

3.6.1 Robustness Against Spoofing Inaccuracies

In §3.5, we directly set spoofed GPS inputs $r_k+\delta_k^a$ based on d and f, and set their uncertainty R_k as the medium value in real-world traces. However, in practice both can have variations due to sensor noises. In this section, we denote the variances to $r_k + \delta_k^a$ as σ_{pos}, and those to R_k as σ_{var}.

Inaccuracy sources and modeling. As specified in our threat model (§3.2), we assume

that the attacker can estimate the victim AD vehicle's real-time positions based on her own position and the distance to the victim. Thus, there are three possible error sources for σ_{pos}: 1) localization error σ_1 in attacker's self-localization process, 2) distance measurement error σ_2 in the measured distance between the attack vehicle and the victim AD vehicle, and 3) GPS receiver error σ_3, i.e., the difference between the position the attacker intended to set and the actual received position at the victim side. Assuming the attacker is equipped with the same sensor set used in an AD system and can run an MSF algorithm of similar quality, σ_1 will be similar to the inaccuracies of BA-MSF algorithm, which is reported as 0.054 meters in [386]. Since LiDAR can be used to measure the distance to the victim, σ_2 is thus the distance measurement error in the LiDAR sensor, which is 0.02 meters as specified in the datasheet according to the LiDAR model used in Apollo [383]. For σ_3, we directly use the positioning error, 0.01 meters, as specified in the datasheet of the GPS model used in Apollo [290]. Assuming that these errors are normally distributed with a zero-mean (common practice in robotics [361]), the combined distribution for σ_{pos} is conforming to $N(0, \sigma_1^2 + \sigma_2^2 + \sigma_3^2) = N(0, 0.058^2)$. For the measurement uncertainty error σ_{var} during spoofing, we measure the distribution of GPS measurement uncertainty in the *ba-local* trace, and take the standard deviation $\sigma_{\text{var}} = 0.008$.

Experimental setup. We apply these error distributions to the FusionRipper attack in *ba-local* using the best attack parameter in *ba-local* with 2-minute minimum attack duration. For each GPS spoofing input, we randomly sample a position error from $N(0, \sigma_{\text{pos}}^2)$ and the error direction from a uniform distribution between 0 to 360 degrees, and apply them to the spoofed input. Similarly, we randomly sample an error value from $N(0, \sigma_{\text{var}}^2)$ and apply it to the measurement uncertainty of each spoofing input. To further explore the impact of these errors, we also apply 2× and 3× amounts of the normal error (σ_{pos} and σ_{var}), in our evaluation. We repeat the experiment 100 times for each error amount.

Results. Fig. 3.11 shows the attack success rates under each error amount. As shown,

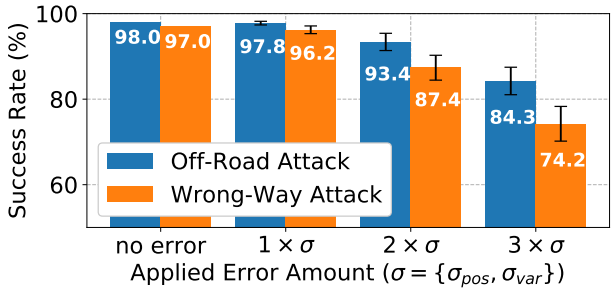

Figure 3.11: Attack success rate for different amounts of spoofing errors. Experiment of each error amount is repeated 100 times.

under normal error amount ($1 \times \{\sigma_{\text{pos}}, \sigma_{\text{var}}\}$), the success rate is only reduced by 0.2% for the off-road attack, and by 0.8% for the wrong-way attack. Even when the error amount is 3× than normal, meaning that the error can be as large as 0.174 meters, the success rate is still 84.3% and 74.2% on average for off-road and wrong-way attacks respectively. This shows that FusionRipper is highly robust to spoofing inaccuracies in practice.

3.6.2 End-to-End Attack Impact Evaluation

In §3.5, we assume the amount of deviation in MSF outputs is the same as the amount of physical position deviations to the center line. In this section, we concretely evaluate this assumption by performing an end-to-end attack impact evaluation with the AD control taking effect.

Evaluation methodology. In this evaluation, we adopt two evaluation methods popularly used in AD industry [167, 96]: *trace based* and *simulation based*. In the trace-based evaluation, we still use the original real-world sensor trace *ba-local*, and synthesize the sensor input changes corresponding to the output of the control module in Apollo. Specifically, the lateral controller in Apollo runs a linear-quadratic regulator algorithm [166] on the lateral deviation in the MSF output, which calculates the amount of steering that will be applied to correct the deviation. We thus mathematically translate such steering into physical position

and heading rate changes (detailed in Appendix B), and add them to the original LiDAR locator position and IMU values to get the changed ones due to AD control. The benefit of this method is that it contains real-world sensor noises, which is the key contributor to the take-over vulnerability (§3.3). However, it does not model more complicated sensing and vehicle motion factors such as raw LiDAR point cloud changes and tire-road frictions, which thus may have limited synthesizing accuracy.

In the simulation-based evaluation, we directly use an AD simulator to dynamically generate raw sensor inputs to Apollo according to its control decisions in the real time, which has more advanced sensor and vehicle motion modelling. However, a common limitation for AD simulators today [146, 238] is that they do not consider generating sensor data with real-world noises. To address this, we model the LiDAR noises as position errors following a normal distribution with a zero mean for each point of the raw LiDAR point cloud generated from the simulator according to the LiDAR datasheet [383].

Experimental setup. In the trace-based evaluation, we run Apollo version 2.5 (the latest version directly compatible with *ba-local*) with the control module enabled on a GPU server, and feed trace *ba-local*. We write a standalone ROS node that feeds the spoofed GPS inputs and also performs the LiDAR locator and IMU input changes described above. For FusionRipper, we use the best attack parameter in *ba-local* with 2-minute minimum attack duration. We do not run the perception module since in Apollo the perception module only outputs detected road obstacles and the system solely relies on the localization module to identify deviations on the road. This is the most popular design modularization for high-level AD systems today [28, 29, 31, 7, 213], which lets the localization module to take charge of all aspects related to vehicle positioning.

In the simulation-based evaluation, we use LGSVL, a production-grade AD simulator that can interface with Apollo version 5.0 [238]. Since Apollo version 5.0 replaces the ROS runtime with Cyber [7], we implement the attack logic and noise modeling in a Cyber node

instead. Different from the trace-based evaluation, we run the simulation on the *complete* Baidu Apollo AD system with all functional modules enabled, i.e., localization, transform, perception, prediction, planning, routing, and control [7]. We simulate two attack scenarios with one attacking to the left of the road and another to the right, where both have concrete safety consequences such as hitting the road barrier or traffic sign.

Trace-based evaluation results. Our results show that FusionRipper achieves 97.0% and 93.9% success rates for off-road and wrong-way attacks respectively, which is only slightly lower than those in the MSF algorithm-only analysis (98.0% and 97.0%). Such slightly effectiveness drop may be due to run-time randomness when running the end-to-end Apollo system since it uses multi-threading when feeding the sensor inputs to BA-MSF.

Simulation-based evaluation results and attack demos. Our simulation results show that FusionRipper can successfully deviate the victim AD vehicle to hit the road barrier or traffic sign even with the complete end-to-end Baidu Apollo AD system operating. We record attack demo videos for these two simulation scenarios, available at our project website `https://sites.google.com/view/cav-sec/fusionripper`. Fig. 3.12 shows a snapshot of the demos. As shown, to correct the MSF output deviation to the right/left of the planned trajectory (i.e., lane center), the AD vehicle in the physical world deviates to the left/right and eventually hit the road barrier or the stop sign.

3.7 Offline Attack Parameter Profiling

Our results so far show that for each trace there always exist an attack parameter combination, i.e., d and f, that can achieve high success rates (§3.5) with high robustness to practical factors (§3.6). However, in §3.5.2 we also observe that such high effectiveness is sensitive to the selection of attack parameters. Thus, it is highly desired if there exists an offline method

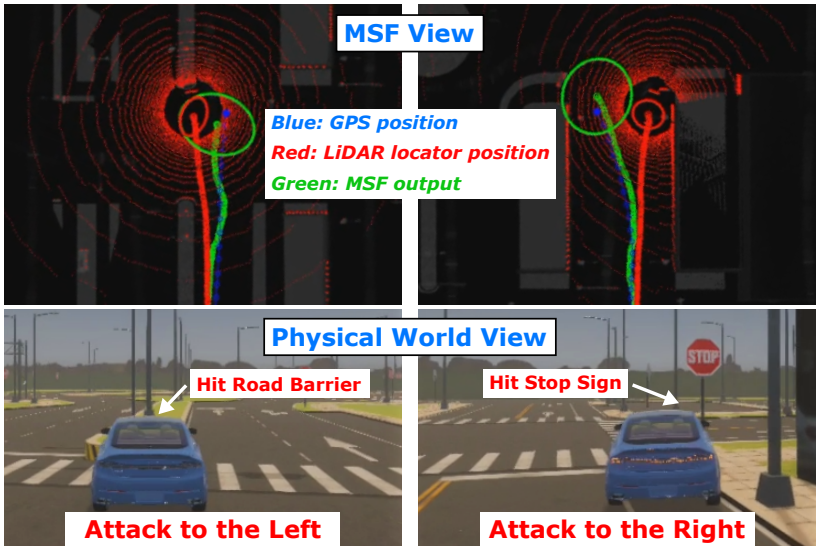

Figure 3.12: Snapshots of our end-to-end attack demos [44]. MSF View: input sensor positions and MSF outputs; Physical World View: victim AD vehicle's physical world position.

that can efficiently identify highly effective attack parameters before the actual attack. In this section, we thus explore the possibility of designing such a method to further improve the practicality of FusionRipper.

3.7.1 Problem Settings and Design

Problem Settings. To find the effective attack parameters offline, we assume that the attacker can perform trials of FusionRipper attacks with different combinations of d and f on AD vehicles of the *same model* as that of the victim AD vehicle, i.e., having the same sensor set, AD system, and vehicle model. This is a realistic assumption since any AD vehicle models developed for commercial purpose need to be mass produced for the ease of management and reducing the development cost for the self-driving taxi or truck services today [48, 41, 23, 9]. For example, Waymo's 20,000 self-driving taxis in Phoenix are deployed with the same sensor suite on the same car model [50], and the same applies to Hyundai's self-driving taxis [21]. In this process, the attack trials can be performed *actively*, by requesting the self-driving taxi or truck services that use the targeted AD vehicle model,

or directly purchasing an AD vehicle of the same model.

In such profiling process, it is necessary to prevent causing obvious safety problems both for the attacker's own safety and for remaining stealthy. Thus, in such offline profiling we choose a *safe profiling design*, which still performs the FusionRipper attack but stops the attack right after the physical-world deviation of the AD vehicle is over a *safe profiling threshold*. This will thus let the non-spoofed GPS and other positioning sources to drag the MSF output deviations back.

Offline profiling algorithm design. Under the problem settings above, our profiling method is designed following a simple strategy: performing attack trials using different combinations of d and f until we find a combination with a sufficiently high success rate. More specifically, the trials are performed for a number of *profiling rounds*. In each round, the attacker picks one combination of d and f and tries it for multiple times. When picking the combinations, the attacker follows the order from the smallest one to the largest one in the parameter space, since larger ones can more easily make the spoofed inputs outliers and thus directly cause attack failure.

Due to the safety requirement, the attacker follows the safe profiling design above, and considers a d and f combination as successful once it reaches the safe profiling threshold. After each profiling round, the attacker can thus obtain a success rate for a d and f combination. Once the success rate of a combination in a round is over a *minimum profiling success rate*, the profiling terminates and such combination is selected for the actual attack. If the attack parameters space is exhausted, the combination with the highest success rate in profiling is selected. The pseudocode of this method is in Algorithm 1.

Algorithm 1 Offline Attack Parameter Profiling

Notations:
AttackTrials(d, f, n, t): Profile n attack trials with parameters d, f, returns the number of trials that have deviations larger than t
N: Number of attack trials in each profiling round
S: Minimum profiling success rate
T: Safe profiling threshold
Output: d, f, cost
Initialize $d, d_{\text{best}} \leftarrow d_{\min}$; $f, f_{\text{best}} \leftarrow f_{\min}$; $\text{SuccRate}_{\text{best}}, \text{cost} \leftarrow 0$

for each $f \leftarrow f_{\min}$ to $f_{\max}$ **do**
  for each $d \leftarrow d_{\min}$ to $d_{\max}$ **do**
    SuccCount $\leftarrow$ AttackTrials(d, f, N, T)
    cost $\leftarrow$ cost $+ N$
    SuccRate $\leftarrow$ SuccCount$/N$
    if SuccRate $\geq S$ **then**
      return d, f, cost
    else
      if SuccRate $>$ $\text{SuccRate}_{\text{best}}$ **then**
        $d_{\text{best}} \leftarrow d$, $f_{\text{best}} \leftarrow f$
        $\text{SuccRate}_{\text{best}} \leftarrow$ SuccRate
      end if
    end if
  end for
end for
return d_{best}, f_{best}, cost

3.7.2 Experiments and Evaluation

Experimental setup. In this section, we use the 5 KAIST traces used in §3.5.2 since this represents the case with attacking the same AD vehicle model (the KAIST traces are collected using the same vehicle on different roads [203]). We split the 5 traces into two sets, with 4 as the *profiling traces*, i.e., representing the attack trials in the offline profiling, and 1 as the *evaluation trace* for evaluating the selected d and f from profiling, i.e., representing the actual attack on the victim AD vehicle. We evaluate all the 5 possible splittings, and then use their average success rate to measure the offline profiling effectiveness. We use the same parameter space as that in §3.5.

Algorithm parameter choices. In the profiling algorithm, there are two configurable parameters: *minimum profiling success rate*, and *safe profiling threshold*. Thus, we first perform experiments to understand how to best configure them. In these experiments, for

each d and f combination we consider all attack starting points in the profiling traces as its corresponding set of attack trials in the profiling algorithm in order to understand general properties of different parameter values.

We first perform experiments by running the profiling algorithm for different minimum profiling success rates without considering safe profiling design. Our results in Fig. 3.13 (a) show that the average success rate of the selected d and f does not change significantly overall. Particularly, it peaks when the minimum profiling success rate is 50% for both attack goals and drops after that, maybe due to the overfitting to the profiling traces.

Next, with 50% as the minimum profiling success rate, we vary the safe profiling threshold, and find that reducing the safe profiling thresholds only slightly changes the average success rate of the selected d and f as shown in Fig. 3.13 (b): the success rate differences between profiling threshold 0.3 and 0.9 meters are less than 4% for both attack goals. In particular, using 0.45 meters as the safe profiling threshold has the overall highest average success rate for both attack goals, which are 90.3% and 84.4% respectively. Such 0.45 meters deviation does not cause the AD vehicle to drive off road on both local roads and highway (Table 3.1). On local roads, it will only cause very slightly lane straddling, and on the highway, it is far from even touching the left or right lane line (both visualized in Fig. A.2 in Appendix). Thus, the attacker can choose to perform such safe profiling on the highway, or on the local roads with light traffic.

Evaluation results. With the algorithm parameter values decided, we then evaluate the algorithm effectiveness and the profiling cost with limited number of attack trials for each combination of d and f in the profiling round. We define profiling cost as the total number of attack trials spent in the profiling algorithm, since in our problem setting each trial corresponds to a self-driving trip the attacker needs to take, e.g., from a targeted self-driving taxi service. For each attack trial, we limit its maximum duration to 90 seconds, which generally covers over 95% of the successful cases according to our earlier evaluation on

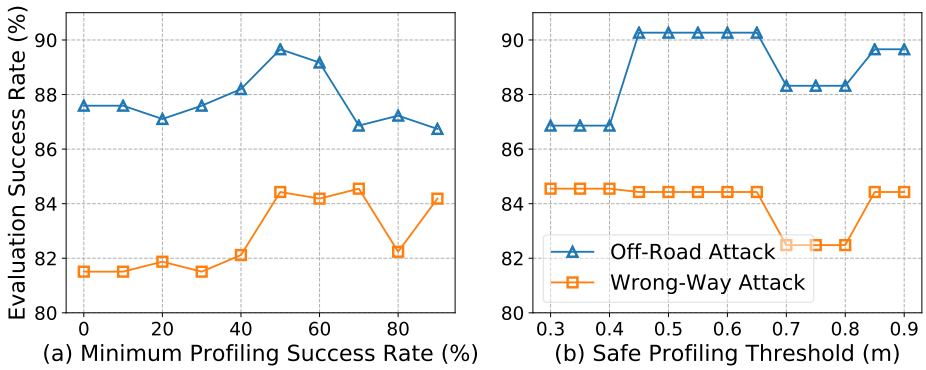

Figure 3.13: Profiling results when using different (a) minimum profiling success rates, and (b) safe profiling thresholds.

attack success time (§3.5.2).

Fig. 3.14 shows the average success rates of the d and f output by the profiling algorithm and the average numbers of 90-sec profiling trips under different numbers of attack trials in each profiling round. In each profiling round, we randomly sample the corresponding number of attack trials from all attack starting points in the profiling traces. As shown, the average success rate increases as the attacker spends more trials in each profiling round since with more trials, the profiled success rate of a d and f combination in a profiling round is statistically closer to the ground truth. Particularly, when the number of trials in each profiling round is 40, our profiling algorithm can find a d and f combination with over 80% average success rate for both off-road and wrong-way attacks (84.2% and 80.7% respectively). In this case, *the profiling cost is only 42 1.5-minute trips on average, which in total is only slightly over 1 hour*. Since the attackers can actively perform such trials, e.g., by requesting self-driving taxi services themselves, finishing this should take at most half a day.

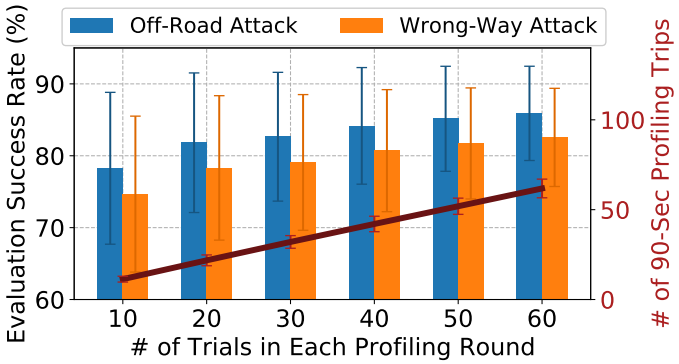

Figure 3.14: Average profiling effectiveness (bar graph) and costs (line graph) under different numbers of attack trials in each profiling round. Each profiling is repeated for 100 times.

3.8 Limitation and Defense Discussions

3.8.1 Limitations of Our Study

Study representativeness. As the first work to study the security of MSF-based AD localization, we choose to focus on the most representative design, KF-based MSF, and the most representative implementation we can find, BA-MSF (representativeness discussed in §2.1.1). However, it is still unclear whether other less common MSF designs (e.g., particle filter based [415]) and outlier detection designs (e.g., expectation-maximization based [363]) can be more secure, which can be potential future work directions.

Attack generality. Although our results have shown the generality of FusionRipper by showing high success rates on 3 different KF-based MSFs (§3.5.4), only one (BA-MSF) of them is production-grade implementation for AD systems. Ideally it is better to evaluate on other production-grade ones, but very unfortunately BA-MSF is the only one that is publicly available so far and it is unlikely for other AD companies to publicly release their implementations in the near future. Thus, due to the lack of information, it is unclear whether other leading AD companies, e.g., Waymo and GM, are vulnerable to our attack. Nevertheless, since BA-MSF is representative both at the design and implementation levels

(§2.1.1) and our attack is general to KF-based MSF by design (§3.3.2), if other AD companies also adopt such a representative design, at least at design level they are also susceptible to the discovered take-over vulnerability. Thus, as the first study, we believe our current discovery and evaluation results can already most generally benefit the understanding of the security property of MSF-based AD localization today.

Attack practicality. We evaluate FusionRipper on real-world traces and under various practical factors such as spoofing inaccuracies and AD control taking effect (§3.6). To further improve the attack practicality, we design an offline attack parameter profiling method that can achieve 84.2% and 80.7% success rates for off-road and wrong-way attacks, with the profiling cost of at most half a day. Nevertheless, due to the cost and legal regulation for GPS spoofing, we did not conduct attack experiments on real-world AD vehicles, which thus can be a valuable future work. Note that GPS spoofing has been proven practical on various end systems [364, 198, 414, 280, 164, 216, 37, 101], including cars such as Tesla cars [37] (§2.1.2). Moreover, in this work, we model GPS spoofing based on attack capabilities shown in prior work [414, 280, 101] to minimize any unrealistic assumptions.

As mentioned in §3.2.2, we assume the attacker owns an AD vehicle and can leverage AD perception algorithms to track the physical position of the victim. Although accurate position-tracking of surrounding obstacles is a basic task for AD, we did not conduct physical-world experiments to confirm this, which is thus left as a valuable future work.

3.8.2 Defense Discussions

In this section, we discuss the potential defense directions against FusionRipper.

Defend against GPS spoofing. Our attack depends on GPS spoofing, so one direct defense direction is to leverage existing GPS spoofing detection or prevention techniques.

Unfortunately, neither GPS spoofing detection nor prevention are fully-solve problems today. On the detection side, numerous techniques have been proposed leveraging signal power monitoring [78, 310, 316], multi-antenna based signal arrival angle detection [263, 310], or crowdsourcing based cross-validation [202]. However, they either can be circumvented by more advanced spoofers [216, 310] or are only applicable to limited domains such as airborne GPS receivers [202]. On the prevention side, cryptographic authentication based civilian GPS infrastructure can fundamentally prevent direct fabrications of GPS signals [310]. However, it requires significant modifications to the existing satellite infrastructure and GPS receivers, and is still vulnerable to replay attacks [298]. Thus, one interesting future work direction is to more concretely understand how effective the latest GPS spoofing defense techniques can be against the current or adapted versions of FusionRipper.

Improve confidence of MSF state and LiDAR locator. Another fundamental defense direction is to improve the positioning confidence of MSF state and LiDAR locator, the two most important factors to the take-over vulnerability in real-world trace (§3.3). Fundamentally, such lacks of confidence in practice result from algorithm inaccuracies and sensor noises (§3.3), and as shown in our analysis, even for the high-end sensors and production-grade LiDAR locator used in AD vehicles today, these inaccuracies and noises are unfortunately large and frequent enough for FusionRipper to exploit. To improve on this, substantial technology breakthrough in sensing and LiDAR-based localization needs to take place. Unfortunately, it is unclear when such breakthrough can take place.

Leverage independent positioning sources (e.g., camera-based lane detection) as fail-safe features for high-level AD localization. Since fundamental defense directions above are not immediately deployable, it is highly desired to discuss the possibility of short-term mitigation solutions. One promising direction is to leverage independent positioning sources to cross-check the localization results and thus serve as *fail-safe* features for AD localization. For example, since both off-road and wrong-way attacks will cause the victim

AD vehicle to deviate from the current lane, they should be detectable by *camera-based lane detection* [186], a mature technology available in many vehicle models today [19]. However, we find that in the high-level AD system design today, such a technology has not been generally considered for fail-safe purposes. For example, the latest release of Baidu Apollo (version 5.5) uses it only for camera calibration [8], while Autoware does not use it at all [6]. This might be because the lane detection output is local positioning within the current lane boundaries, and thus cannot be directly used for comparison against global positioning from MSF. However, the vulnerability discovered in this work strongly motivates the need for considering adding such kind of fail-safe features in future AD localization, at least for *anomaly detection*. Note that more investigations are needed to understand how effective and robust such kind of fail-safe features can be in the defense. For example, when camera-based lane detection is applied for anomaly detection, the precision/recall rates need to be further explored since it needs to carefully consider (1) AD vehicles legitimately deviating from current lane due to routing requirements, and (2) lane line scratches or incompleteness. Moreover, camera-based lane detection itself is vulnerable to physical-world attacks [59, 327].

Note that even if such fail-safe features can perform perfect attack detection, our attack still causes denial-of-service of the victim's global localization function, which can render the victim in unsafe scenarios, e.g., stopping in the middle of highway lanes, since the victim can neither correctly reach the destination nor safely locate the road shoulder to pull over. Thus, a more useful defense direction is to *correct* the attacked localization results. However, so far the global positioning accuracy of cameras is unsatisfying for high-level AD localization, especially along the longitudinal direction (forward/backward) since only the stop lines can be used as features [127, 233]. This is why LiDAR locator is used more predominantly in high-level AD localization (§2.1.1). Moreover, such correction is yet another multi-sensor fusion problem and thus is still fundamentally vulnerable to the take-over vulnerability discovered in this work (§3.3). Thus, how to leverage other independent positioning sources to effectively perform such correction under our attack is still an open research challenge, which can be a

valuable future work direction.

3.9 Summary

In this chapter, we perform the first security study on MSF-based localization in high-level AD settings under GPS spoofing. We discover a take-over vulnerability that can fundamentally defeat the MSF design principle, and design FusionRipper, a novel and general attack that opportunistically captures and exploits it. Our evaluation on real-world traces shows that FusionRipper can achieve over 97% and 91.3% success rates in all traces for off-road and wrong-way attacks. Such high effectiveness is also found highly robust to various practical factors. We also design an offline method that can identify effective attack parameters within at most half a day. We also discuss both long-term and short-term defenses directions, and identify that a promising mitigation is to use camera-based lane detection as a fail-safe feature, which has not been generally considered for such purpose today. As the first study on AD localization security, we hope that our findings and insights can bring immediate attention and inspire the development of effective defenses considering the critical role of localization for safe and correct driving.

Chapter 4

Security Challenges in AD Perception

4.1 Region-of-Interest Attack on Traffic Light Detection

4.1.1 Introduction

In the automotive industry, a revolution is taking place: the rise of Autonomous Driving (AD) Vehicles. AD vehicles have been demonstrated to lower transportation costs and energy consumption, improve travel convenience and comfort, and reduce traffic accidents and congestion [91]. However, the AD system, which serves as the "brain" of an AD vehicle to make driving decisions, may have vulnerabilities that could lead to severe security threats to road safety. For example, vulnerabilities in the localization module, whose outputs are critical for route planning and navigation, can be exploited by attackers to manipulate the routes of AD vehicles [258]. In fact, attacks on the localization will not only affect AD vehicles' route planning and navigation but also have direct effects in other modules such as the perception module to cause false detection.

The perception module processes sensor inputs to perceive the surrounding obstacles and traffic lights, which are necessary for planning a safe driving path and making the correct driving decision. Typically, the perception sensors (i.e., camera, LiDAR, radar, ultrasonic sensor) in the AD system [7, 213] have wide ranges of views. For example, AD vehicles are usually equipped with high-resolution cameras that are capable of detecting obstacles as far as 100 meters or more [7]. However, in this work, we discover an interesting design consideration, which is common in AD systems, that enables an attacker to *blind or misguide the perception without tampering the perception sensor inputs themselves*.

The root cause for such a vulnerability is the design of Region-of-Interests (ROIs). ROI is a strategy commonly employed in AD perception that utilizes information from the localization to *narrow* the detection scope in the sensor inputs [7, 213]. It can effectively reduce the computation overhead by running detection algorithms on smaller input dimensions and filter detection noises by preventing ambiguous detection, e.g., when multiple traffic lights exist in the camera view. Despite the benefits in improving the detection efficiency and accuracy, the accuracy of ROI mainly depends on the localization results–a wrong localization would result into a wrong ROI, which in turn causes the perception module to look at a wrong area in the sensor input.

For AD systems, cameras are especially critical for traffic light (TL) detection since they are the only sensors that are able to accurately detect TL colors. Thus, as the first study, we start from the TL detection and leverage the existing GPS spoofing attacks [216, 335] on localization to demonstrate attack consequences of such ROI attacks in an end-to-end AD system. Attack demos showing the end-to-end attack consequences are at `https://sites.google.com/view/roiattack`.

Considering the severity of the attack, we hope this work can bring immediate attention to the AD system developers for more robust ROI designs. In summary, this work makes the following contributions:

- We perform the first security analysis on the ROI design in the perception module in AD systems and identify a design-level vulnerability that allows the attacker to fool AD perception using GPS spoofing.
- We design a concrete ROI attack targeting the TL detection in AD perception. Results show that our attack is able to achieve a 100% success rate in causing the victim AD vehicle to run red lights or denial-of-service.

4.1.2 Threat Model and Attack Goal

Threat model. Similar to the threat model used in prior works [335, 414], we assume a car-following model where the attacker launches GPS spoofing attack while tailgating the victim AD vehicle. We assume the attacker can arbitrarily manipulate the victim's localization outputs under GPS spoofing, which is valid for AD vehicles using only GPS for localization, such as Baidu Apollo in GPS mode and Tesla. We also assume the attacker can track the physical position of the victim AD vehicle, which is feasible if the attacker's vehicle is also an AD vehicle [335].

Attack goal. The goal of our attack is to influence the position of ROI in the perception module via GPS spoofing, and thus cause the victim AD vehicle to detect a wrong TL to run the red light or fail to detect any TLs to stop at the green light (i.e., denial-of-service or DoS).

4.1.3 Attack Insight and Design

Attack insight. Although from high-level design, the localization and perception modules appear to be independent of each other, the perception module in fact heavily relies on the localization, especially for the ROI logic in TL detection as mentioned in §2.1. Since

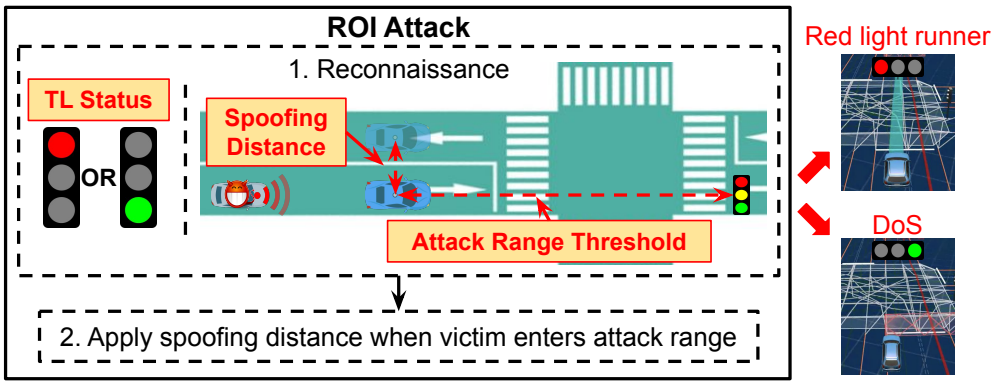

Figure 4.1: Illustration of the ROI attack for TL detection.

the localization output directly determines the ROI that the TL detection pipeline will be performed on, any errors in the localization would result in a wrong ROI and may cause incorrect TL detection.

Attack design. To perform the ROI attack on TL detection, the attacker launches GPS spoofing when the victim AD vehicle is in front of an intersection. When the victim AD vehicle drives towards the intersection, a longitudinal error in the localization would naturally have a smaller effect on the ROI position compared to a lateral deviation. Thus, to maximize the effect on the ROI, we design the attack as spoofing a *lateral* distance away from the physical position of the victim AD vehicle, e.g., the spoofing distance shown on the Fig. 4.1.

Fig. 4.1 illustrates the ROI attack for TL detection, which is composed of a reconnaissance stage and a spoofing stage. In the reconnaissance stage, the attacker examines the current TL status and determines the best attack parameters to be applied. Depending on the TL status, the attacker will use different spoofing distances to achieve the desired attack goals. For example, when the TL is red, the attacker needs to spoof a distance such that the ROI includes another green TL in the camera view to cause false detection. Alternatively, if the TL is green, the attacker can simply spoof a distance large enough such that the ROI contains no TLs, which would trigger a safe stop in the planning since the AD vehicle knows from the HD map that there is a TL in the front but fails to detect one. In addition,

we set an *attack range threshold* for determining the timing to launch the spoofing. If the attacker starts spoofing when the victim is too far or too close to the intersection, the victim might have already been deviated out of the road boundary that prevented it from reaching the intersection or have already detected the correct TL color and committed the correct driving decision. After that, the attacker closely monitors the victim's position and spoofs the corresponding lateral distance when the victim enters the attack range.

4.1.4 Evaluation

Experimental setup. Due to the high cost of evaluating self-driving algorithms on real AD vehicles, we follow the common practice for AD vehicle testing and perform a simulation-based evaluation, in which we run Baidu Apollo in an industry-grade AD simulator, LGSVL [238]. To facilitate AD simulation, LGSVL provides photo-realistic simulation environments with diverse road structures and a wide range of vehicle models. In our evaluation, we use the Shalun map and Lincoln MKZ vehicle. The Shalun map models a common two-lane road with multiple intersections along the road. We use Lincoln MKZ since it contains the compatible sensor configurations for Baidu Apollo. We simulate our attack on Baidu Apollo version 5.0, which is the latest version fully supported by LGSVL.

To simulate the attack consequences, we create two concrete attack scenarios: *red light detection* and *green light detection*. Specifically, for red light detection, we set the front TL in the first intersection to red and the back TL in the second intersection to green; for green light detection, we set the front TL to green and the back TL to red. We have confirmed that in benign driving, the AD vehicle will always correctly detect the front TLs, and make the correct driving decisions, i.e., stop before the intersection or drive through the intersection.

Evaluation metric. In our evaluation, we explore the attack effectiveness of the ROI attack under different attack range thresholds and spoofing distances, aiming to find the parameters

that can achieve the highest attack effectiveness. We define a successful attack case as the victim AD vehicle mis-detects the front TL and commits the wrong driving decisions, i.e., running the red light or stopping in front of the green light. Since both Baidu Apollo and LGSVL involve random factors such as messaging delays, we calculate a success rate by repeating the simulation for 5 times for each attack parameter.

Attack construction. For the ease of evaluation, we implement the attack logic as an independent module in Baidu Apollo to receive original GPS inputs from LGSVL. If the victim AD vehicle is within the attack range threshold, we apply the spoofing distance to the original GPS positions. After that, we publish the spoofed GPS inputs to the localization module.

Results. As mentioned in §4.1.3, the red light detection scenario has a more restricted spoofing distance requirement since the attacker needs to spoof the GPS such that the ROI covers the back green TL but not the front red TL. Thus, we start by exploring the attack parameters for this scenario. The left figure in Fig. 4.2 shows the attack success rates when using different attack range thresholds, i.e., the distance from the AD vehicle to the TL that triggers the attack. During the experiments, a spoofing distance of 3 meters is applied to the victim's GPS inputs. As shown, the best attack range threshold falls between 24 and 28 meters. When using a small attack range threshold (e.g., 22 meters), the attack fails because the victim is too close to the intersection, and it already executed the stop decision based on the TL detection prior to the attack. On the other hand, if the attack range threshold is too large (e.g., over 28 meters), the victim might already be deviated out of road boundary since the AD system is constantly correcting any deviation between the localization and the planned trajectory [335].

The right figure in Fig. 4.2 shows the attack success rates when using different spoofing distances. Based on the previous experiments on the attack range threshold, we launch the attack when the victim AD vehicle is 26 meters away from the TL. As shown in Fig. 4.2, a

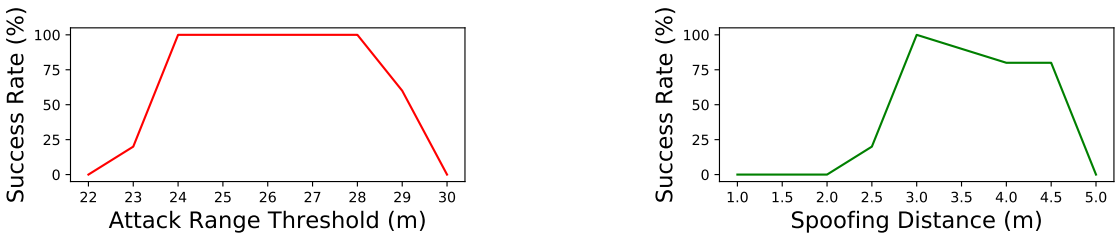

Figure 4.2: Attack success rates under different attack range thresholds (left) and spoofing distances (right).

spoofing distance of smaller than or equal to 2 meters is too small to move the red TL out of the ROI, so the victim can still detect it. On the other hand, when the spoofing distance is larger than 4.5 meters, both front and back TLs are moved out of the ROI, causing the victim to fail to detect any TL and simply stop in front of the intersection. When the spoofing distance is 3–4.5 meters, the attack success rate is at least 80%, where the front red TL is moved out of the ROI, but the back green TL is still in and thus detected by the victim AD vehicle, causing it to continue driving forward to run the red light. In particular, a 100% attack success rate is achieved when using a spoofing distance of 3 meters.

In the green light detection scenario, instead of controlling the ROI to be at a particular position, the attacker can simply spoof the ROI to the area without any TLs. In such a case, the TL detector will report "unknown" as no TLs exist in the ROI. Consequently, the planning module in the AD system issues a stop decision conservatively. This results in a DoS attack since the victim stops at the green TL. Similarly, we evaluate this scenario and found that our ROI attack can achieve a 100% success rate when using a spoofing distance of 5 meters.

Attack demos. We create two attack demos to illustrate the severe consequences of the ROI attack. Fig. 4.3 and Fig. 4.4 show the snapshots when attacking the red and green light detections, respectively. The left sub-figures show the camera images with the ROI annotations, and the right sub-figures show the corresponding driving decisions in Baidu Apollo. As shown, our attack can successfully fool the victim AD vehicle to detect a wrong

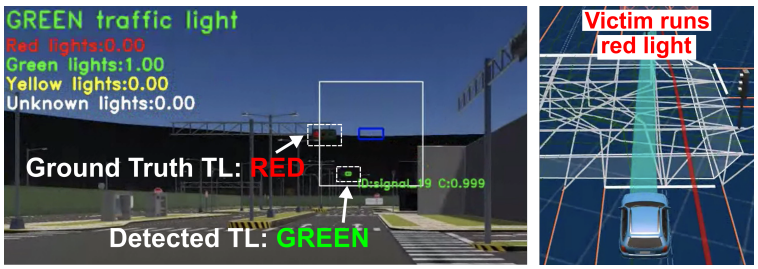

Figure 4.3: Snapshot of attacking the red light detection. The green TL at the next intersection is falsely detected by the AD vehicle due to the incorrect ROI.

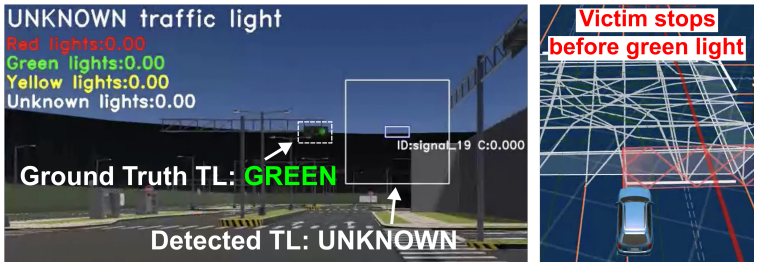

Figure 4.4: Snapshot of attacking the green light detection. The TL detector failed to detect any light in the ROI, causing the victim to stop at the stop line despite the light is green.

TL or fail to detect any TLs due to the ROI position shifts caused by GPS spoofing. Such an attack poses great dangers to road safety since it can cause the AD vehicle to violate traffic rules and may even lead to car crashing consequences, e.g., when another vehicle fails to yield in time when the victim AD vehicle is running the red light. Demo videos with and without attacks are available at `https://sites.google.com/view/roiattack`.

4.1.5 Limitations and Future Work

As the first study of ROI attacks in AD perception using GPS spoofing, we start by attacking AD system with a GPS-based localization, where GPS is the only positioning resource. Indeed, for GPS-based localization, it is possible to simply spoof the GPS to a road without any TLs, such that the victim is not in a state where it realizes that it should sense a TL, resulting in red light running. However, this is not the case for MSF-based localization, which is predominantly adopted in today's Level-4 AD vehicles, since GPS alone can no

longer dictate the localization output. Although MSF is considered to be more robust against GPS spoofing, prior work has demonstrated that MSF-based localization can be deviated by as large as 10 meters using GPS spoofing [335]. In the future, we plan to incorporate the MSF attack to improve the practicality of the ROI attack. Another limitation is that we only evaluate the attack on one AD system, i.e., Baidu Apollo [7]. Although Baidu Apollo implements a representative TL ROI design and is already in production, it is necessary to also evaluate on other TL ROI implementations (e.g., Autoware [213]) to show the generality of our attack. We leave this as a future work. Besides, since the attack parameters are highly dependent on factors such as victim's speed and road shape, more diverse driving scenarios are needed to evaluate the applicability of our attack. In addition, currently the best attack parameters are found via exhaustive search. In the next step, we plan to use the victim AD vehicle/TL positions and camera configurations to calculate the best spoofing distances to be applied in different scenarios.

4.1.6 Summary

In this work, we perform the first security analysis on perception ROI design and take the TL detection as a case study. We discover that using GPS spoofing can cause the AD vehicle to recognize a wrong TL or fail to detect any TLs. Our study shows that strategic GPS spoofing can achieve a 100% success rate to fool TL detection in the AD vehicle, leading to red-light running or denial-of-service consequences, which can be detrimental to road safety. The future of AD vehicles is bright, but more security investigations are still needed in the AD systems. We hope our research can bring more attention to the security threats to AD vehicles, improving AD systems for safer AD vehicles for the public.

4.2 Dirty Road Patch Attack on Automated Lane Centering

4.2.1 Introduction

Automated Lane Centering (ALC) is a Level-2 driving automation technology that automatically steers a vehicle to keep it centered in the traffic lane [326]. Due to its high convenience for human drivers, today it is widely available on various vehicle models such as Tesla, GM Cadillac, Honda Accord, Toyota RAV4, Volvo XC90, etc. While convenient, such a system is highly security and safety critical: When the ALC system starts to make wrong steering decisions, the human driver may not have enough reaction time to prevent safety hazards such as driving off road or colliding into vehicles in adjacent lanes. Thus, it is imperative and urgent to understand the security property of ALC systems.

In an ALC system, the most critical step is *lane detection*, which is generally performed using a front camera. So far, Deep Neural Network (DNN) based lane detection achieves the highest accuracy [54] and is adopted in the most performant production ALC systems today such as Tesla Autopilot [33]. Recent works show that DNNs are vulnerable to physical-world adversarial attacks such as malicious stickers on traffic signs [158, 157, 120, 420]. However, these methods cannot be directly applied to attack ALC systems due to two main design challenges. First, in ALC systems, the physical-world attack generation needs to handle *inter-dependencies* among camera frames due to attack-influenced vehicle actuation. For example, if the attack deviates the detected lane to the right in a frame, the ALC system will steer the vehicle to the right accordingly. This causes the following frames to capture road areas more to the right, and thus directly affect their attack generation. Second, the optimization objective function designs in prior works are mainly for image classification or object detection models and thus aim at changing class or bounding box

probabilities [158, 420]. However, attacking lane detection requires changing the *shape* of the detected traffic lane, making it difficult to directly apply prior designs.

To fill this critical research gap, in this work we are the first to systematically study the security of DNN-based ALC systems in their designed operational domains (i.e., roads with lane lines) under physical-world adversarial attacks. Since ALC systems assume a fully-attentive human driver prepared to take over at any time [34, 326], we identify the attack goal as not only causing the victim to drive out of the current lane boundaries, but also achieving it shorter than the average driver reaction time to road hazard. This thus directly breaks the design goal of ALC systems and can cause various types of safety hazards such as driving off road and vehicle collisions.

Targeting this attack goal, we design a novel physical-world adversarial attack method on ALC systems, called *DRP (Dirty Road Patch) attack*, which is the first to systematically address the design challenges above. First, we identify *dirty road patches* as a novel and domain-specific attack vector for physical-world adversarial attacks on ALC systems. This design has 2 unique advantages: (1) Road patches can appear to be legitimately deployed on traffic lanes in the physical world, e.g., for fixing road cracks; and (2) Since it is common for real-world roads to have dirt or white stains, using similar dirty patterns as the input permutations can allow the malicious road patch to appear more normal and thus stealthier.

With this attack vector, we then design systematic malicious road patch generation following an optimization-based approach. To efficiently and effectively address the first design challenge without heavyweight road testing or simulations, we design a novel method that combines vehicle motion model and perspective transformation to dynamically synthesize camera frame updates according to attack-influenced vehicle control. Next, to address the second design challenge, one direct solution is to design the objective function to directly change the steering angle decisions. However, we find that the lateral control step in ALC that calculates steering angle decisions are generally not differentiable, which makes it dif-

ficult to effectively optimize. To address this, we design a novel lane-bending objective function as a differentiable surrogate function. We also have domain-specific designs for attack robustness, stealthiness, and physical-world realizability.

We evaluate our attack method on a production ALC system in OpenPilot [131], which is reported to have close performance to Tesla Autopilot and GM Super Cruise [64, 69, 56]. We perform experiments on 80 attack scenarios from real-world driving traces, and find that our attack is highly effective with over 97.5% success rates for all scenarios, and less than 0.903 sec average success time, which is substantially lower than 2.5 sec, the average driver reaction time (§4.2.2). This means that even for a fully-attentive driver who can take over as soon as the attack starts to take effect, the average reaction time is still not enough to prevent the damage.

To understand the potential safety impacts, we further conduct experiments using software-in-the-loop simulation in a production-grade simulator. Results show that our attack can successfully cause a victim running a production ALC to hit the highway concrete barrier or a truck in the opposite direction with 100% success rates.

In summary, this work makes the following contributions:

- We are the first to systematically study the security of DNN-based ALC in the designed operational domains under physical-world adversarial attacks. We formulate the problem with a safety-critical attack goal, and a novel and domain-specific attack vector, dirty road patches.
- To systematically generate attack patches, we adopt an optimization-based approach with 2 major novel and domain specific designs: motion model based input generation, and lane-bending objective function.
- We perform evaluation on a production ALC using real-world driving traces. The

results show that our attack is highly effective with ≥97.5% success rates and ≤0.903 sec average success time, which is substantially lower than the average driver reaction time.

- To understand the safety impacts, we conduct experiments using (1) software-in-the-loop simulation, and (2) attack trace injection in a real vehicle. The results show that our attack can cause a 100% collision rate in different scenarios, including when tested with safety features such as AEB.

Code and data release. *Our code and data for the attack and evaluations are available at our project website [17].*

4.2.2 Attack Formulation and Challenge

Attack Goal and Incentives

In this work, we consider an attack goal that directly breaks the design goal of ALC systems: causing the victim vehicle a lateral deviation (i.e., deviating to the left or right) large enough to drive out of the current lane boundaries. Meanwhile, since ALC systems assume a fully-attentive human driver who is prepared to take over at any moment [34, 326], such deviation needs to be achieved fast enough so that the human driver cannot react in time to take over and steer back. Table 4.1 shows concrete values of these two requirements for successful attacks on highway and local roads respectively. In the table, the required deviations are calculated based on representative vehicle and lane widths in the U.S., and the required success time is determined using commonly-used average driver reaction time to road hazards, which is detailed in Appendix C.

Targeted scenario: Free-flow driving. Our study targets the most common driving scenario for using ALC systems: *free-flow* driving scenarios [419], in which a vehicle has at least

Table 4.1: Required deviations and success time for successful attacks on ALC systems on highway and local roads. Detailed calculations and explanations are in Appendix C.

Road Type	Required Lateral Deviation	Required Success Time
Highway	0.735 m	<2.5 sec (average driver reaction time to road hazard)
Local road	0.285 m	

5–9 seconds clear headway [104] and thus can drive freely without considering the front vehicle [419].

Safety implications. The attack goal above can directly cause various safety hazards in the real world: (1) *Driving off road*, which is a direct violation of traffic rules [14] and can cause various safety hazards such as hitting road curbs or falling down the highway cliff. (2) *Vehicle collisions*, e.g., with vehicles parked on the roadside, or driving in adjacent or opposite traffic lanes on a local road or a two-lane undivided highway. Even with obstacle or collision avoidance, these collisions are still possible for two reasons. First, today's obstacle and collision avoidance systems are not perfect. For example, a recent study shows that the AEB (Automatic Emergency Braking) systems in popular vehicle models today fail to avoid crashes 60% of the time [18]. Second, even if they can successfully perform emergency stop, they cannot prevent the victim from being hit by other vehicles that fail to yield on time.

Threat Model

We assume that the attacker can obtain the same ALC system as the one used by the victim to get a full knowledge of its implementation details. This can be done through purchasing or renting the victim vehicle model and reverse engineering it, which has already been demonstrated possible on Tesla Autopilot [59]. Moreover, there exist production ALC systems that are open sourced [131]. We also assume that the attacker can obtain a motion model [315] of the victim vehicle, which will be used in our attack generation process (§4.2.3). This is a realistic assumption since the most widely-used motion model (used by us in §4.2.3)

only needs vehicle parameters such as steering ratio and wheelbase as input [315], which can be directly found from vehicle model specifications. We assume the victim drives at the speed limit of the target road, which is the most common case for free-flow driving. In the attack preparation time, we assume that the attacker can collect the ALC inputs (e.g., camera frames) of the target road by driving the victim vehicle model there with the ALC system on.

Design Challenges

Compared to prior works on physical-world adversarial attacks on DNNs, we face 3 unique design challenges:

C1. Lack of legitimately-deployable attack vector in the physical world. To affect the camera input of an ALC system, it is ideal if the malicious perturbations can appear legitimately around traffic lane regions in the physical world. To achieve high legitimacy, such perturbations also must not change the original human-perceived lane information. Prior works use small stickers or graffiti in physical-world adversarial attacks [158, 420, 59]. However, directly performing such activities to traffic lanes in public is illegal [53]. In our problem setting, the attacker needs to operate in the middle of the road when deploying the attack on traffic lanes. Thus, if the attack vector cannot be disguised as legitimate activities, it becomes highly difficult to deploy the attack in practice.

C2. Camera frame inter-dependency due to attack-influenced vehicle actuation. In real-world ALC systems, a successful attack on one single frame can barely cause any meaningful lateral deviations due to the steering angle change limit at the vehicle actuation step (§2.1.1). For example, for the vehicle models with 0.25° angle change limit per control loop, even if a successful attack on a single frame causes a very large steering angle decision at MPC output (e.g., 90°), it can only cause at most 1.25° actuated steering angle changes before the next

frame comes, which can only cause up to *0.3-millimeter* lateral deviations at 45 mph (∼72 km/h).

Thus, to achieve our attack goal in §4.2.2, the attack must be *continuously effective on sequential camera frames* to increasingly reach larger actuated steering angles and thus larger lateral deviations per frame. In this process, due to the dynamic vehicle actuation applied by the ALC system, the attack effectiveness for later frames are directly dependent on that for earlier frames. For example, if the attack successfully deviates the detected lane to the right in a frame, the ALC system will steer the vehicle to the right accordingly. This causes the following frames to capture road areas more to the right, and thus directly affect their attack generation. There are prior works considering attack robustness across sequential frames, e.g., using EoT [90, 108] and universal perturbation [241], but none of them consider frame inter-dependencies due to attack-influenced vehicle actuation in our problem setting.

C3. Lack of differentiable objective function design for LD models. To systematically generate adversarial inputs, prior works predominately adopt optimization-based approaches, which have shown both high efficiency and effectiveness [352, 158, 113, 151]. However, the objective function designs in these prior works are mainly for image classification [158, 108] or object detection [120, 157, 420] models, which thus aim at decreasing class or bounding box probabilities. However, as introduced in §2.1.1, LD models output detected lane line curves, and thus to achieve our attack goal the objective function needs to aim at changing the *shape* of such curves. This is substantially different from decreasing probability values, and thus none of these existing designs can directly apply.

Closer to our problem, prior works that attack end-to-end autonomous driving models [301, 362, 124, 422] directly design their objective function to change the final steering angle decisions. However, as described in §2.1.1, state-of-the-art LD models do not directly output steering angle decisions. Instead, they output lane line curves and rely on the lateral control step to compute the final steering angle decisions. However, many steps in the lateral control

module, e.g., the desired driving patch calculation and the MPC framework, are generally not differentiable to the LD model input (i.e., camera frames), which makes it difficult to effectively optimize.

4.2.3 Dirty Road Patch Attack Design

In this work, we are the first to systematically address the design challenges above by designing a novel physical-world attack method on ALC, called *Dirty Road Patch (DRP) attack*.

Design Overview

To address the 3 design challenges in §4.2.2, our DRP attack method has the following novel design components:

Dirty road patch: Domain-specific & stealthy physical-world attack vector. To address challenge **C1**, we are the first to identify *dirty road patch* as an attack vector in physical-world adversarial attacks. This design has 2 unique advantages. First, road patches can appear to be legitimately deployed on traffic lanes in the physical world, e.g., for fixing road cracks. Today, deploying them is made easy with adhesive designs [58] as shown in Fig. 4.5. The attacker can thus take time to prepare the attack in house by carefully printing the malicious input perturbations on top of such adhesive road patches, and then pretend to be road workers like those in Fig. 4.5 to quickly deploy it when the target road is the most vacant, e.g., in late night, to avoid drawing too much attention.

Second, since it is common for real-world roads to have dirt or white stains such as those in Fig. 4.5, using similar dirty patterns as the input perturbations can allow the malicious road patch to appear more normal and thus stealthier. To mimic the normal dirty patterns, our design only allows color perturbations on the gray scale, i.e., black-and-white. To avoid

Figure 4.5: Illustration of our novel and domain-specific attack vector: Dirty Road Patch (DRP).

changing the lane information as discussed in §4.2.2, in our design we (1) require the original lane lines to appear exactly the same way on the malicious patch, if covered by the patch, and (2) restrict the brightness of the perturbations to be strictly lower than that of the original lane lines. To further improve stealthiness, we also design parameters to adjust the perturbation size and pattern, which are detailed in §4.2.3.

So far, none of the popular production ALC systems today such as Tesla, GM, etc. [34, 60, 2, 40, 68, 65, 62, 61, 63] identify roads with such dirty road patches as driving scenarios that they do not handle, which can thus further benefit the attack stealthiness.

Motion model based input generation. To address the strong inter-dependencies among the camera frames (**C2**), we need to dynamically update the content of later camera frames according to the vehicle actuation decisions applied at earlier ones in the attack generation process. Since adversarial attack generation typically takes thousands of optimization iterations [112, 262], it is practically highly difficult, if not impossible, to drive real vehicles on the target road to obtain such dynamic frame update in every optimization iteration. Another idea is to use vehicle simulators [238, 147], but it requires the attacker to first create a high-definition 3D scene of the target road in the real world, which requires a significant amount of hardware resource and engineering efforts. Also, launching a vehicle simulator in

each optimization iteration can greatly harm the attack generation speed.

To efficiently and effectively address this challenge, we combine *vehicle motion model* [315] and *perspective transformation* [183, 354] to dynamically synthesize camera frame updates according to a driving trajectory simulated in a lightweight way. This method is inspired by Google Street View [83] that synthesizes 360° views from a limited number of photos utilizing perspective transformation. Our method only requires one trace of the ALC system inputs (i.e., camera frames) from the target road without attack, which can be easily obtained by the attacker (§4.2.2).

Optimization-based DRP generation. To systematically generate effective malicious patches, we adopt an optimization-based approach similar to prior works [352, 158]. To address challenge **C3**, we design a novel lane-bending objective function as a differentiable surrogate that aims at changing the derivatives of the desired driving path before the lateral control module, which is equivalent to change the steering angle decisions at the lateral control design level. Besides this, we also have other domain-specific designs in the optimization problem formulation, e.g., for a differentiable construction of the curve fitting process, malicious road patch robustness, stealthiness, and physical-world realizability.

Fig. 4.6 shows an overview of the malicious road patch generation process, which is detailed in the following sections.

Motion Model based Input Generation

In Fig. 4.6, step ①–⑦ belong to the motion model based input generation component. As described earlier in §4.2.3, the input to this component is a trace of ALC system inputs such as camera frames from driving on the target road without attack. In ①, we apply *perspective transformation*, a widely-used computer vision technique that can project an image view from a 3D coordinate system to a 2D plane [183, 354]. Specifically, we apply it

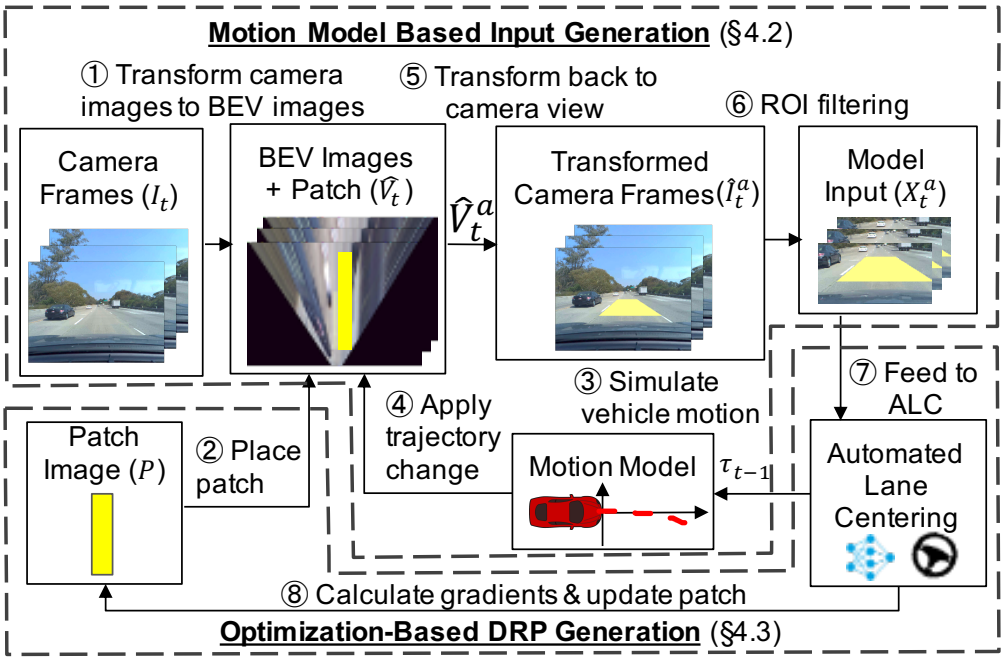

Figure 4.6: Overview of our DRP (Dirty Road Patch) attack method. ROI: Region of Interest; BEV: Bird's Eye View.

to the original camera frames from the driver's view to obtain their Bird's Eye View (BEV) images. This transformation is highly beneficial since it makes our later patch placement and attack-influenced camera frame updates much more natural and thus convenient. We denote this as $V_t := \text{BEV}(I_t)$, where I_t and V_t are the original camera input and its BEV view respectively at frame t. This process is invertible, i.e., we can also obtain I_t with $\text{BEV}^{-1}(V_t)$.

Next, in ②, we obtain the generated malicious road patch image P from the optimization-based DRP generation step (§4.2.3) and place it on V_t to obtain the BEV image with the patch, denoted as $\widehat{V}_t := \Lambda(V_t, P)$. To achieve consistent patch placements in the world coordinate across frames, we calculate the *pixel-meter relationship*, i.e., the number of pixels per meter, in BEV images based on the driving trace of the target road. With this, we can place the patch in each frame precisely based on the driving trajectory changes across frames.

Next, we compute the vehicle moving trajectory changes caused by the placed malicious

road patch, and reflect such changes in the camera frames. We represent the vehicle moving trajectory as a sequence of vehicle states $S_t := [x_t, y_t, \beta_t, v_t], (t = 1, ..., T)$, where x_t, y_t, β_t, v_t are the vehicle's 2D position, heading angle, and speed at frame t, and T is the total number of frames in the driving trace. Thus, the trajectory change at frame t is $\delta_t := S_t^a - S_t^o$, where S_t^a and S_t^o are vehicle states with and without attack respectively.

To calculate δ_t caused by the attack effect at the frame $t-1$, we need to know the attack-influenced vehicle state S_t^a. To achieve that, we use a *vehicle motion model* to simulate the vehicle state S_t^a by feeding the steering angle decision τ_{t-1} from the lateral control step in the ALC system (§2.1.1) given the attacked frame at $t-1$ and the previous vehicle state S_{t-1}^a, denoted as $S_t^a := \text{MM}(S_{t-1}^a, \tau_{t-1})$. A vehicle motion model is a set of parameterized mathematical equations representing the vehicle dynamics and can be used to simulate its driving trajectory given the speed and actuation commands. In this process, we set the vehicle speed as the speed limit of the target road as described in our threat model (§4.2.2). In our design, we adopt the kinematic bicycle model [225], which is the most widely-used motion model for vehicles [22, 225, 392].

With δ_t, in ④ we then apply affine transformations on the BEV image $\widehat{V}_t$ to obtain the attack-influenced one $\widehat{V}_t^a$, denoted as $\widehat{V}_t^a := T(\widehat{V}_t, \delta_t)$. Fig. 4.7 shows an example of the shifting and rotation $T(\cdot)$ in the BEV, which synthesizes a camera frame with the vehicle position shifted by 1 meter and rotated by 10° to the right. Although it causes some distortion and missing areas on the edge, the ROI area (red rectangle), i.e., the LD model input, is still complete and thus sufficient for our purpose. Since the ROI area is typically focused on the center and much smaller than the raw camera frame (§2.1.1), our method can successfully synthesize multiple complete LD model inputs from only 1 ALC system input trace.

Next, in ⑤, we obtain the attack-influenced camera frame at the driver's view $\widehat{I}_t^a$, i.e., the direct input to ALC, by projecting $\widehat{V}_t^a$ back using $\widehat{I}_t^a := \text{BEV}^{-1}(\widehat{V}_t^a)$. Next, in ⑥, the ROI filtering is used to extract the model input $X_t^a := \text{ROI}(\widehat{I}_t^a)$. X_t^a and vehicle state

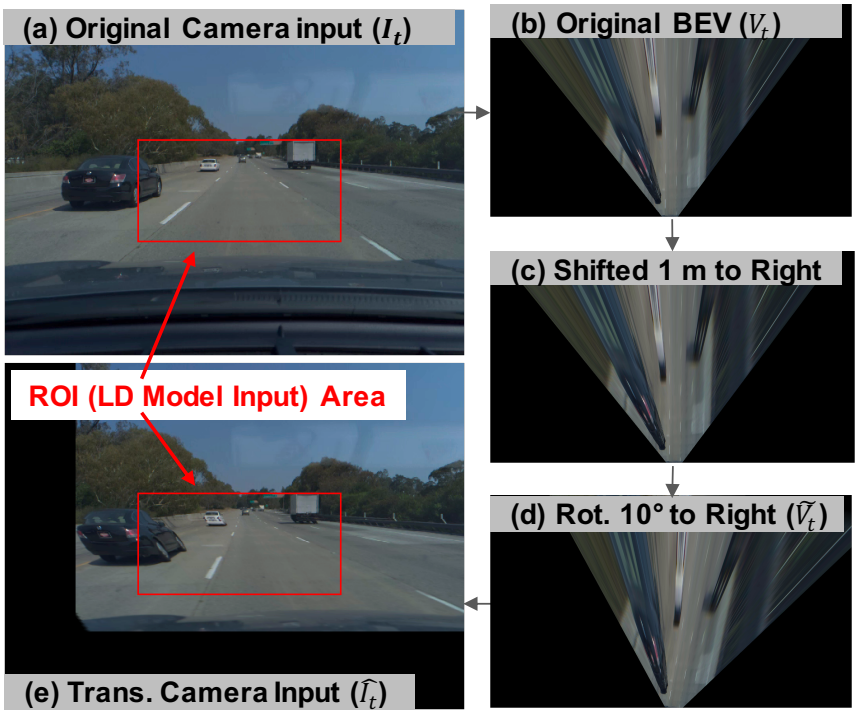

Figure 4.7: Motion model based input generation from original camera input.

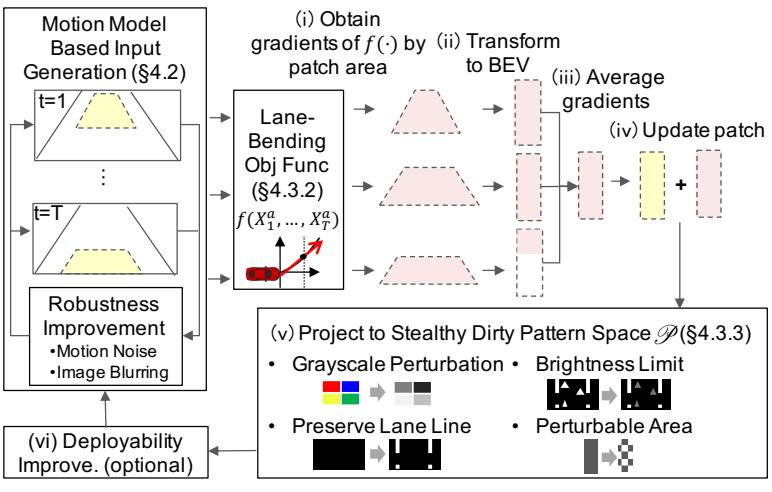

Figure 4.8: Iterative optimization process design for our optimization-based DRP generation.

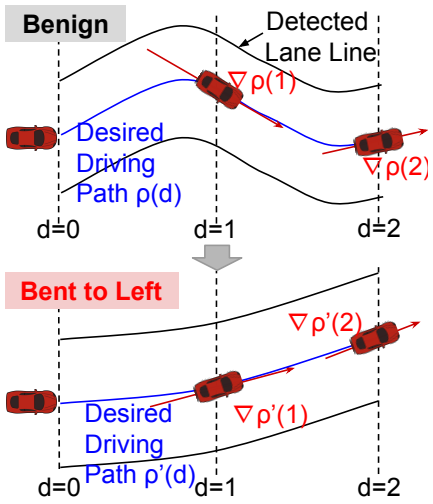

Figure 4.9: Lane bending effect of our objective function.

S_t^a are then fed to ALC system in ⑦ to obtain the steering angle decision τ_t, denoted as $\tau_t := \text{ALC}(X_t^a, S_t^a)$. Step ③–⑦ are then iteratively applied to obtain $\widehat{I}_{t+1}^a, \widehat{I}_{t+2}^a, ...$ one after one until all the original frames are updated to reflect the moving trajectory changes caused by P. These updated attack-influenced inputs are then fed to the optimization-based DRP generation component, which is detailed next.

Optimization-Based DRP Generation

In Fig. 4.6, step ⑧ belongs to the optimization-based road path generation component. In this step, we design a domain-specific optimization process on the target ALC system to systematically generate the malicious dirty road patch P.

DRP attack optimization problem formulation. We formulate the attack as the following optimization problem:

$$\min \quad \mathscr{L} \tag{4.1}$$

$$\text{s.t.} \quad X_t^a = \text{ROI}(\text{BEV}^{-1}(T(\Lambda(V_t, P), S_t^a - S_t^o))) \quad (t = 1, ..., T) \tag{4.2}$$

$$\tau_t^a = \text{ALC}(X_t^a, S_t^a) \qquad (t = 1, ..., T) \tag{4.3}$$

$$S_{t+1}^a = \text{MM}(S_t^a, \tau_t^a) + \epsilon_t \qquad (t = 1, ..., T-1) \tag{4.4}$$

$$S_1^a = S_1^o \tag{4.5}$$

$$P = \text{BLUR}(\text{FILL}(B) + \Delta) \tag{4.6}$$

$$\Delta \in \mathscr{P} \tag{4.7}$$

where the $\mathscr{L}$ in Eq. 4.1 is an objective function that aims at deviating the victim out of the current lane boundaries as fast as possible (detailed in §4.2.3). Eq. 4.2–4.5 have been described in §4.2.3. In Eq. 4.6, the patch image $P \in \mathbb{R}^{H \times W \times C}$ consists of a base color $B \in \mathbb{R}^C$ and the perturbation $\Delta \in \mathbb{R}^{H \times W \times C}$, where W, H, and C are the patch image width, height, and the number of color channels respectively. We select an asphalt-like color as the base

color B since the image is designed to mimic a road patch. Function FILL: $\mathbb{R}^C \rightarrow \mathbb{R}^{H \times W \times C}$ fills B to the entire patch image. Since we aim at generating perturbations that mimic the normal dirty patterns on roads, we restrict Δ to be within a stealthy road pattern space $\mathscr{P}$, which is detailed in §4.2.3. We also include a noise term ϵ_t in Eq. 4.4 and an image blurring function BLUR$(\cdot)$ in Eq. 4.6 to improve the patch robustness to vehicle motion model inaccuracies and camera image blurring.

Optimization process overview. Fig. 4.8 shows an overview of our iterative optimization process design. Given an initial patch image P, we obtain the model input $X_1^a, ..., X_T^a$ from the motion model based input generation process. In step (i), we calculate the gradients of the objective function with respect to $X_1^a, ..., X_T^a$, and only keep the gradients corresponding to the patch areas. In step (ii), these gradients are projected into the BEV space. In step (iii), we calculate the average BEV-space gradients weighted by their corresponding patch area sizes in the model inputs. This step involves an approximation of the gradient of BEV$^{-1}(\cdot)$, which are detailed in Appendix D. Next, in step (iv), we update the current patch with Adam [221] using the averaged gradient as the gradient of the patch image. In step (v), we then project the updated patch into the stealthy road pattern space $\mathscr{P}$. This updated patch image is then fed back to the motion model based input generation module, where we also add robustness improvement such as motion noises and image blurring. We terminate this process when the attack-introduced lateral deviations obtained from the motion model are large enough.

Lane-bending objective function design. As discussed in §4.2.3, directly using steering angle decisions as $\mathscr{L}$ makes the objective function non-differentiable to $X_1^a, ..., X_T^a$. To address this, we design a novel lane-bending objective function $f(\cdot)$ as a differentiable surrogate function. In this design, our key insight is that at the design level, the lateral control step aims at making steering angle decisions that follow a *desired driving path* in the middle of the detected left and right lane line curves from the lane detection step (§2.1.1). Thus, changing

the steering angle decisions is equivalent to changing the derivatives of (or "bending") such desired driving path curve. This allows us to design $f(\cdot)$ as:

$$f(X_1^a, ..., X_T^a) = \sum_{t=1}^{T} \sum_{d \in D_t} \nabla \rho_t(d; \{X_j^a | j \leq t\}, \theta) + \lambda ||\Omega_t(X_t^a)||_p \tag{4.8}$$

where $\rho_t(d)$ is a parametric curve whose parameters are decided by (1) both the current and previous model inputs $\{X_j^a | j \leq t\}$ due to frame inter-dependencies (§4.2.2), and (2) the LD DNN parameters θ. D_t is a set of curve point index $d = 0, 1, 2, ...$ for the desired driving path curve at frame t. λ is the weight of the p-norm regularization term, designed for stealthiness (§4.2.3). We then can define $\mathscr{L}$ in Eq. 4.1 as $f(\cdot)$ and $-f(\cdot)$ when attacking to the left and right. Fig. 4.9 illustrates this surrogate function when attacking to the left. As shown, by maximizing $\nabla \rho_t(d)$ at each curve point in Eq. 4.8, we can achieve a "lane bending" effect to the desired driving path curve. Since the direct LD output is lane line points (§2.1.1) but $\rho_t(\cdot)$ require lane line curves, we further perform a differentiable construction of curve fitting process (Appendix D).

Designs for dirty patch stealthiness. To mimic real-world dirty patterns like in Fig. 4.5, we have 4 stealthiness designs in stealthy road pattern space $\mathscr{P}$ in Eq. 4.7:

Grayscale perturbation. Real-world dirty patterns on the road are usually created by dust or white stains (Fig. 4.5), and thus most commonly just appear white. Thus, we cannot allow perturbations with arbitrary colors like prior works [420]. Thus, our design restricts our perturbation Δ in the grayscale (i.e., black-and-white) by only allowing increase the Y channel in the YCbCr color space [182], denoted as $\Delta_Y \geq 0$.

Preserving original lane line information. We preserve the original lane line information by drawing the same lane lines as the original ones on the patch (if covered by the patch). Note that without this our attack can be easier to succeed, but as discussed in §4.2.2, it is

much more preferred to preserve such information so that the attack deployment can more easily appear as legitimate road work activities and the deployed patch is less likely to be legitimately removed.

Brightness limits. While the dirty patterns are restricted to grayscale, they are still the darker, the stealthier. Also, to best preserve the original lane information, the brightness of the dirty patterns should not be more than the original lane lines. Thus, we (1) add the p-norm regularization term in Eq. 4.8 to suppress the amount of Δ_Y, and (2) restrict $B_Y + \Delta_Y < \text{LaneLine}_Y$, where B_Y and LaneLine_Y are Y channel values for the base color and original lane line color respectively.

Perturbation area restriction. Besides brightness, also the fewer patch areas are perturbed, the stealthier. Thus, we define Perturbable Area Ratio (PAR) as the percentage of pixels on P that can be perturbed. Thus, when PAR=30%, 70% pixels on P will only have the base color B.

4.2.4 Attack Methodology Evaluation

Targeted ALC system. In our evaluation, we perform experiments on the production ALC system in OpenPilot [131], which follows the state-of-the-art DNN-based ALC system design (§2.1.1). OpenPilot is an open-source production Level-2 driving automation system that can be easily installed in over 80 popular vehicle models (e.g., Toyota, Cadillac, etc.) by mounting a dashcam. We select OpenPilot due to its (1) *representativeness*, since it is reported to have close performance to Tesla Autopilot and GM Super Cruise and better than many others [64, 69, 56], (2) *practicality*, from the large quantity and diversity of vehicle models it can support [131], and (3) *ease to experiment with*, since it is the only production ALC system that is open sourced. In this work, we mainly evaluate on the lane detection model in OpenPilot v0.7.0, which is released in Dec. 2019.

Figure 4.10: Driver's view at 2.5 sec (average driver reaction time to road hazards [12]) before our attack succeeds under different stealthiness levels in local road scenarios. Inset figures are the zoomed-in views of the malicious road patches.

Figure 4.11: Real-world dirty road patterns.

Figure 4.12: Stop sign hiding and appearing attacks [420].

Evaluation dataset. We perform experiments using the comma2k19 dataset [329], which contains over 33 hours driving traces between California's San Jose and San Francisco in a Toyota RAV4 2017 driven by human drivers. These traces are collected using the official OpenPilot dashcam device, called EON. From this dataset, we manually look for short free-flow driving periods to make road patch placement convenient. In total, we obtain 40 eligible short driving clips, 10 seconds each, with half of them on the highway, and half on local roads. For each driving clip, we consider two attack scenarios: attack to the left, and to the right. Thus, in total we evaluate 80 different attack scenarios.

Attack Effectiveness

Evaluation methodology and metrics. We evaluate the attack effectiveness using the evaluation dataset described above. For each attack scenario, we generate an attack road patch, and use the motion model based input generation method in §4.2.3 to simulate the

vehicle driving trajectory influenced by the malicious road patch. To judge the attack success, we use the attack goal defined in §4.2.2 and concrete metrics listed in Table 4.1, i.e., achieving over 0.735 m and 0.285 m lateral deviations on highway and local road scenarios respectively within the average driver reaction, 2.5 sec. We measure the achieved deviation by calculating the lateral distances at each time point between the vehicle trajectories with and without the attack, and use the earliest time point to reach the required deviation to calculate the success time.

Since ALC systems assume a human driver who is prepared to take over, it is better if the malicious road patch can also look stealthy enough at 2.5 sec (driver reaction time) before the attack succeeds so that the driver won't be alerted by its looking and decide to take over. Thus, in this section, we also study the stealthiness of the generated road patches. Specifically, we quantify their perturbation degrees using the average pixel value changes from the original road surface in $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_{\text{inf}}$ distances [247, 89].

Experimental setup. For each scenario in the evaluation dataset, we manually mark the road patch placement area in the BEV view of each camera frame based on the lane width and shape. To achieve consistent road patch placements in the world coordinate across a sequence of frames, we calculate the number of pixels per meter in the BEV images and adjust the patch position in each frame precisely based on the driving trajectory changes across consecutive frames. The road patch sizes we use are 5.4 m wide, and 24–36 m long to ensure at least a few seconds of visible time at high speed. The patches are placed 7 m far from the victim at the starting frame. For stealthiness levels, we evaluate the $\mathcal{L}_2$ regularization coefficient $\lambda = 10^{-2}, 10^{-3}$, and 10^{-4}, with PAR set to 50%. According to Eq. 4.8, a larger λ value means more suppression of the perturbation, and thus should lead to a higher stealthiness level. For the motion model, we directly use the vehicle parameters (e.g., wheelbase) of Toyota RAV4 2017, the vehicle model that collects the traces in our dataset.

Table 4.2: Attack success rate and time under different stealthiness levels. Larger λ means stealthier. Average success time is calculated only among the successful cases. Pixel $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_{inf}$ are the average pixel value changes from the original road surface in the RGB space and normalized to $[0, 1]$.

Stealth. Level λ	Succ. Rate	Succ. Time (s)	Pixel $\mathcal{L}_1$	Pixel $\mathcal{L}_2$	Pixel $\mathcal{L}_{inf}$
10^{-2}	97.5%	0.903	0.018	0.045	0.201
10^{-3}	100%	0.887	0.033	0.066	0.200
10^{-4}	100%	0.886	0.071	0.109	0.200

Results. As shown in Table 4.2, our attack has high effectiveness ($\geq$97.5%) under all the 3 stealthiness levels. Fig. 4.10 shows the malicious road patch appearances at different stealthiness levels from the driver's view at 2.5 seconds before our attack succeeds. As shown, even for the lowest stealthiness level ($\lambda = 10^{-4}$) in our experiment, the perturbations are still smaller than some real-world dirty patterns such as the left one in Fig. 4.11. In addition, the perturbations for all these 3 stealthiness levels are a lot less intrusive than those in previous physical-world adversarial attacks in the image space [420], e.g., in Fig. 4.12. Among the successful cases, the average success time is all under 0.91 sec, which is substantially lower than 2.5 sec, the required success time. This means that even for a fully attentive human driver who is always able to take over as soon as the attack starts to take effect, the average reaction time is still far from enough to prevent the damage.

Robustness to run-time driving trajectory and angle deviations. The run-time victim driving trajectories and angles will be different from the motion model predicted ones in attack generation time due to run-time driving dynamics. To evaluate attack robustness against such deviations, we use (1) 4 levels of vehicle position shifting at each vehicle control step in attack evaluation time, and (2) 27 vehicle starting positions to create a wide range of approaching angles and distances to the patch, e.g., from (almost) the leftmost to the rightmost position in the lane. Our attack is shown to maintain a high effectiveness ($\geq$ 95% success rate) even when the vehicle positions at the attack evaluation time has 1m shifting on average from those at the attack generation time at each control step.

Physical-World Realizability Evaluation

In physical world, there are 3 main practical factors that can affect the attack effectiveness: (1) the lighting condition, (2) printer color accuracy, and (3) camera sensing capability. Thus, in this section we perform experiments to understand the physical-world attack realizability against these 3 main practical factors.

Evaluation methodology: miniature-scale experiments. To perform the DRP attack, a real-world attacker can pretend to be road workers and place the malicious road patch on public roads. However, due to the access limit to private testing facilities, we cannot do so ethically and legally on public roads with a real vehicle. Thus, we try our best to perform such evaluation by designing a *miniature-scale experiment*, where the road and the malicious road patch are first physically printed out on papers and placed according to the physical-world attack settings but in miniature scale. Then the real ALC system camera device is used to get camera inputs from such a miniature-scale physical-world setting. Such miniature-scale evaluation methodology can capture all the 3 main practical factors in the physical-world attack setting, and thus can sufficiently serve for the purpose of this evaluation.

Experimental setup. As shown in Fig. 4.13, we create a miniature-scale road by printing a real-world high-resolution BEV road texture on multiple ledger-size papers and concatenating them together to form a long straight road. In the attack evaluation, we create the miniature-scale malicious road patch using the same method, and place it on top of the miniature-scale road following our DRP attack design. We mount EON, the official OpenPilot dashcam device, on a tripod and face it to the miniature-scale road. The road size, road patch size, and the EON mounting position are carefully calculated to represent OpenPilot installed on a Toyota RAV4 driving on a standard 3.6 m wide highway road at 1:12 scale. We also create different lighting conditions with two studio lights. The patch size is set to represent a 4.8 m wide and 12 m long one in the real world scale.

Evaluation metric. Since the camera is mounted in a static position, we evaluate the attack effectiveness directly using the steering angle decision at the frame level instead of the lateral deviation used in previous sections. This is equivalent from the attack effectiveness point of view since the large lateral deviation is essentially created by a sequence of large steering angle decisions at the frame level. Specifically, we first find the camera frame that has the same relative position between the camera and the patch as that in the miniature-scale experimental setup. Then we compare its *designed* steering angle at the attack generation time and its *observed* steering angle that the ALC system in OpenPilot intends to apply to the vehicle in the miniature-scale experiment. Thus, the more similar these two steering angles are, the higher realizability our attack has in the physical world.

Results. Fig. 4.14 shows a visualization of the lane detection results of the benign and attacked scenarios in the miniature-scale experiment using the OpenPilot's official visualization tool. As shown, in the benign scenario, both detected lane lines align accurately with the actual lane lines, and the desired driving path is straight as expected. However, when the malicious road patch is placed, it bends the detected lane lines significantly to the left and causes the desired driving path to be curving to the left, which is exactly the designed attack effect of our lane-bending objective function (§4.2.3). In this case, the designed steering angle is 23.4° to the left at the digital attack generation time, and the observed one in the physical miniature-scale experiment is 24.5° to the left, which only differs by 4.7%. In contrast, in the benign scenario the observed steering angle for the same frame is 0.9° to the right.

Robustness under different lighting conditions. We repeat this experiment under 12 lighting conditions ranging from 15 lux (corresponding to sunset/sunrise) to 1210 lux (corresponding to midday of overcast days). The results show that the same attack patch above is able to maintain a desired steering angle of 20-24° to the left under all 12 lighting conditions, which are all significantly different from the benign scenario (0.9° to the right).

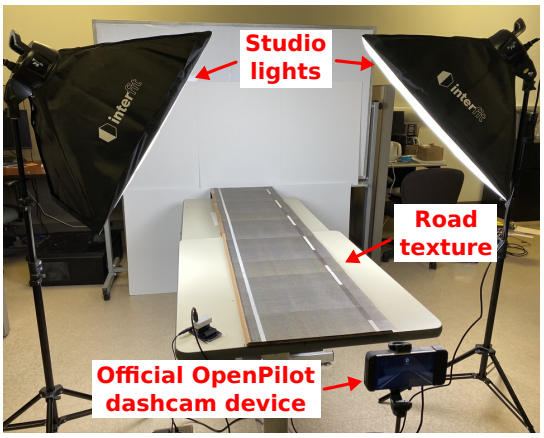

Figure 4.13: Miniature-scale experiment setup. Road texture/patch are printed on ledger-size papers.

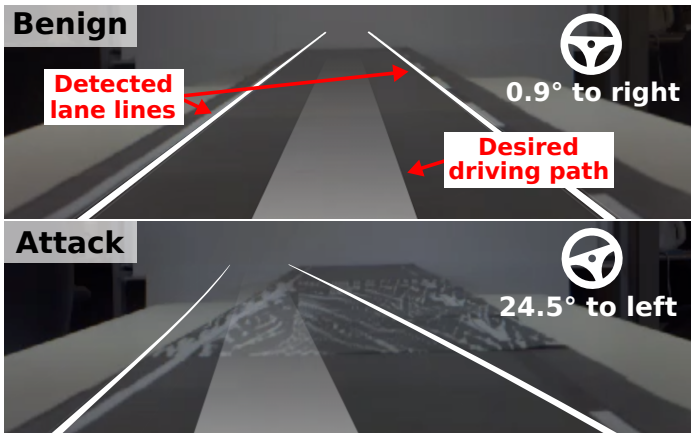

Figure 4.14: Lane detection and steering angle decisions in benign and attacked scenarios in the miniature-scale experiment.

Robustness to different viewing angles. We evaluate the attack robustness from 45 different viewing angles created by different distances to the patch and lateral offsets to the lane center. Our results show that our attack always achieves over 23.4° to the left from all viewing angles. We record videos in which we dynamically change viewing angles in a wide range while showing real-time lane detection results under attack, available at `https://sites.google.com/view/cav-sec/drp-attack/`.

4.2.5 Software-in-the-Loop Simulation

To understand the safety impact, we perform software-in-the-loop evaluation of our attack on LGSVL, a production-grade autonomous driving simulator [238].

Evaluation scenarios and setup. We construct 2 attack scenarios for highway and local road settings respectively, as shown in Fig. 4.15. For the former, we place a concrete barrier on the left, and for the latter, we place a truck driving on an opposite direction lane. The attack goals are to hit the concrete barrier or the truck. We perform evaluation on OpenPilot v0.6.6 with the Toyota RAV4 parameters. We generate the adversarial patch as 5.4 m wide and 70 m long, which is placed in the simulation environment by importing the patch image

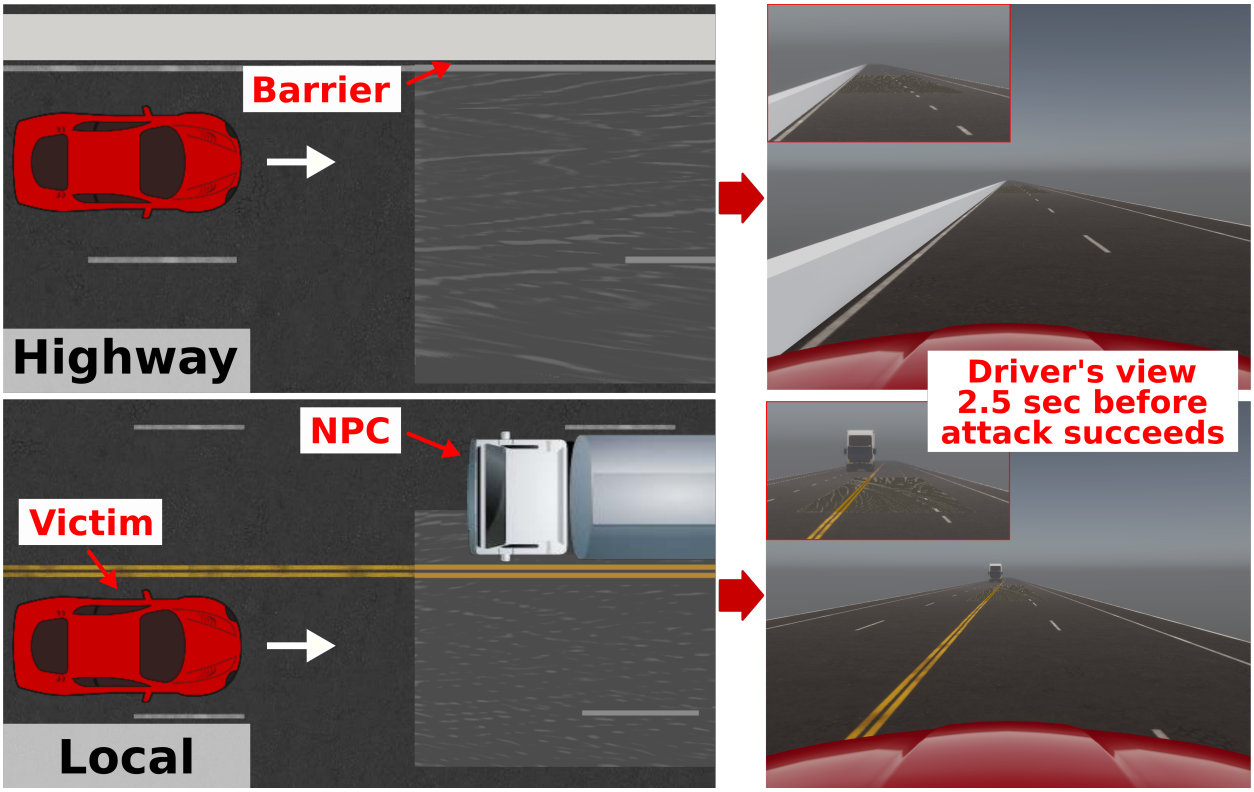

Figure 4.15: Software-in-the-loop simulation scenarios and driver's view 2.5 sec before attack succeeds.

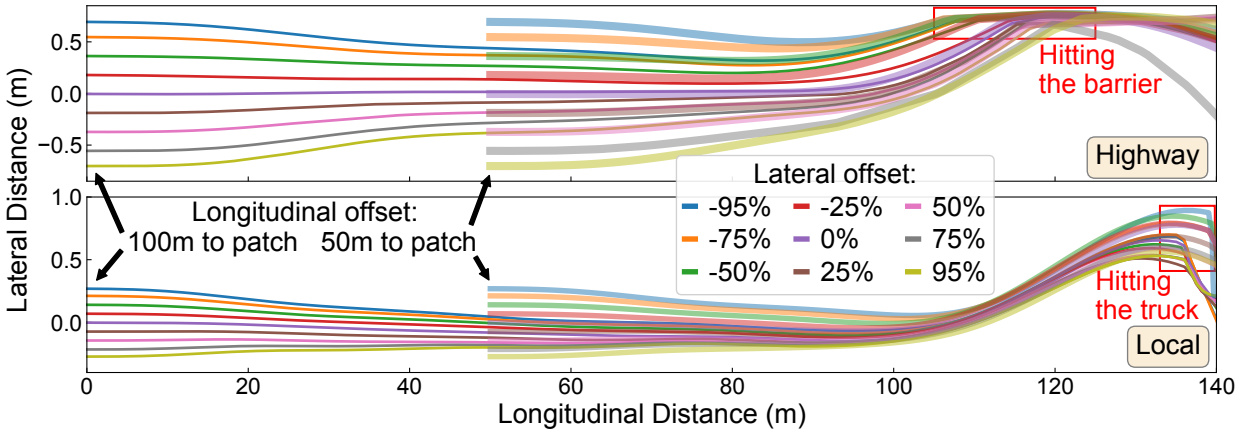

Figure 4.16: Victim driving trajectories in the software-in-the-loop evaluation from 18 different starting positions for highway and local road scenarios. Lateral offset values are percentages of the maximum in-lane lateral shifting from lane center; negative and positive signs mean left and right shifting.

into Unity. To evaluate the attack effectiveness from different victim approaching angles, for each scenario we evaluate the same patch from 18 different starting positions, created from the combinations of 2 longitudinal distances to the patch (50 and 100 m) and 9 lateral offsets (from -95% to 95%) as shown in Fig. 4.16. The patch is visible at all these starting positions. We repeat 10 times for each starting position in each scenario.

Results and video demos. Our attack achieves 100% success rates from all 18 starting positions in both highway and local road scenarios. Fig. 4.16 shows the averaged vehicle trajectories from each starting positions. As shown, the vehicle always first drives toward the lane center since the ALC system tries to correct the initial lateral deviations. After that, the patch starts to take effect, and causes the vehicle to deviate to the left significantly and hit the barrier or truck. We record demo videos at `https://sites.google.com/view/cav-sec/drp-attack/`. In the highway scenario, after the victim hits the concrete barrier, it bounces away quickly due to the abrupt collision. For local road, the victim crashes to the front of the truck, causing both the victim and truck to stop. This suggests that the safety impacts of our attack can be severe.

4.2.6 Defense Discussion

Machine learning model level defenses. In the recent arms race between adversarial machine learning attacks and defenses, numerous defense/mitigation techniques have been proposed [245, 403, 406, 178, 262, 314, 232, 129]. However, so far none of them studied LD models.

Sensor/Data fusion based defenses. Besides securing LD models, another direction is to fuse camera-based lane detection with other independent sensor/data sources such as LiDAR and High Definition (HD) map [20]. For example, LiDAR can capture the tiny laser reflection differences for lane line markings, and thus is possible to perform lane detection [92]. However, while LiDARs are commonly used in high-level (e.g., Level-4) AD systems such as Google Waymo [57] that provide self-driving taxi/truck, so far they are not generally used in production low-level (e.g., Level-2) AD such as ALC, e.g., Tesla, GM Cadillac, Toyota RAV4, etc. [33, 60, 2, 40]. This is mainly because LiDAR is quite costly for vehicle models sold to individuals.

Another possible fusion source is lane information from a pre-built HD map of the targeted road, which can be used to cross-check with the run-time detected lane lines to detect our attack. However, this requires ALC providers to collect and maintain accurate lane line information for each road, which can be time consuming, costly, and also hard to scale. Even with such map, a follow-up research question is how to effectively detect the attack without raising too many false alarms, since mismatched lane information can also occur in benign cases due to (1) vehicle position and heading angle inaccuracies when localized on the HD map, e.g., due to sensor noises in GPS and IMU [386, 236], and (2) normal-case LD model inaccuracies.

4.2.7 Summary

In this work, we are the first to systematically study the security of DNN-based ALC in its designed operational domains under physical-world adversarial attacks. With a novel attack vector, dirty road patch, we perform optimization-based attack generation with novel input generation and objective function designs. Evaluation on a production ALC using real-world traces shows that our attack has over 95% success rates with success time substantially lower than average driver reaction time, and also has high robustness, generality, physical-world realizability, and stealthiness. We further conduct experiments using both simulation and a real vehicle, and find that our attack can cause a 100% collision rate in different scenarios. We also evaluate and discuss possible defenses. Considering the popularity of ALC and the safety impacts shown in this work, we hope that our findings and insights can bring community attention and inspire follow-up research.

Chapter 5

Defense Opportunity against AD Localization Attacks

5.1 Introduction

Recently, high-level Autonomous Driving (AD) vehicles [326], e.g., Level-4 ones, are gradually becoming part of the transportation system by providing commercial services such as self-driving taxis [87, 47], buses [11, 200], and trucks [43, 204]. In particular, AD companies such as Waymo and Baidu are already offering commercial RoboTaxi services without safety drivers [87, 222], and more others are performing tests on public roads [220, 219]. To achieve high driving automation, the *high-level AD system* (the "*brain*") in such a vehicle needs to localize itself with *centimeter-level* accuracy on the map [236, 319, 155] to ensure safe and correct driving. Thus, today's industry-grade high-level AD systems predominantly adopt a Multi-Sensor Fusion (MSF) based localization design, which combines sensor inputs, typically GPS, LiDAR, and IMU, for overall higher accuracy and robustness in practice [386, 170, 340, 29, 28, 31].

Due to the reliance on sensor inputs, AD localization is inherently vulnerable to sensor spoofing attacks, in particular GPS spoofing [335, 37], a long-existing security problem that is fundamentally difficult in both prevention and detection in practice [310, 335]. Although the MSF-based design is generally more robust against such single-source sensor attacks, recent work [335] find that state-of-the-art MSF algorithms are still vulnerable to strategic GPS spoofing attacks due to non-deterministic and practical factors such as sensor noises and algorithm inaccuracies. To leverage such non-deterministic vulnerabilities, the authors devise a lateral-direction localization attack named *FusionRipper* to *opportunistically* inject lateral deviations in the MSF localization outputs, which will be translated into lateral deviations in the physical world by the AD control. Such lateral-direction localization attack is especially safety-critical in the AD context due to the potential consequences of road departure [159].

So far, no software-based defenses have been proposed to defend against such latest lateral-direction localization attack in high-level AD systems. The closed-related ones are the recent *physical-invariant* based defenses for small robotic vehicles such as drones and rovers [313, 126]. Such defenses estimate system states (i.e., vehicle positions) based on control commands and use them to validate GPS signals. While these works show high effectiveness for small robotic vehicles, we find that they have limited effectiveness in the AD context (evaluated as a baseline in §5.5.2), because (1) vehicle driving motions in the real-world are more diverse and complex (e.g., commonly have high-speed or curvy-road driving), and thus harder to model accurately, and (2) attack deviation goals can be much smaller while still being safety-critical, e.g., even less than 0.5-meter lateral deviations can cause lane departure. Moreover, these works did not consider the *attack response* step after detection, which is especially critical for high-level AD systems due to the complex driving environment and the absence of onboard human drivers. A few recent works considered such attack response designs, e.g., attack recovery upon attack detection [125, 416]. However, they rely on similar state estimation models as above to replace the attacked sensors during the recovery period, which thus suffer from the same motion model accuracy limitations in AD context, and also

counted on human operators to take over as soon as possible since such state estimations cannot replace physical sensors for a prolonged duration due to drifting [125]. Last but not least, they assume an effective attack detection in place, which does not yet exist in for high-level AD localization.

Compared to small robotic vehicles, the AD context may also have its unique defense opportunity for lateral-direction localization attacks, for example *Lane Detection (LD)* [186, 297], which is directly related to the attack goals since it can measure the vehicle's physical lateral deviation in the ego lane in the real time. Today, LD is already widely used in low-level AD localization (e.g., for automated lane centering). However, due to its fundamental limit in achieving effective global localization (§5.3), it is currently *not* used for high-level AD localization purposes. While less suitable for accuracy purposes, so far no prior works have explored its potential for *defense purposes* in high-level AD localization.

In this work, we thus perform the first concrete exploration of LD as a domain-specific defense opportunity for lateral-direction localization attacks in high-level AD systems. We start by systematically analyzing its high-level defense potential, and find that such an LD-based defense strategy has various design-level benefits such as generality to lateral-direction attacks, technology maturity, direct deployability, and independence to existing attacks. One potential downside is the lack of defense capability when lane line markings are not available (e.g., in intersections), but since the latest lateral-direction attack design is fundamentally opportunistic, the attacker *cannot* deterministically control the triggering of a desired deviation only at road regions without lane line markings. In fact, we find that a defense coverage of the road regions with lane line marking can already provide protection for *99.2%* of such opportunistic attack attempts (§5.3).

Motivated by such multi-dimensional defense potentials, we design the first domain-specific LD-based defense approach called LD^3 (Lane Detection based Lateral-Direction Localization attack Defense), which is capable of both real-time attack *detection* and *response*. To use LD

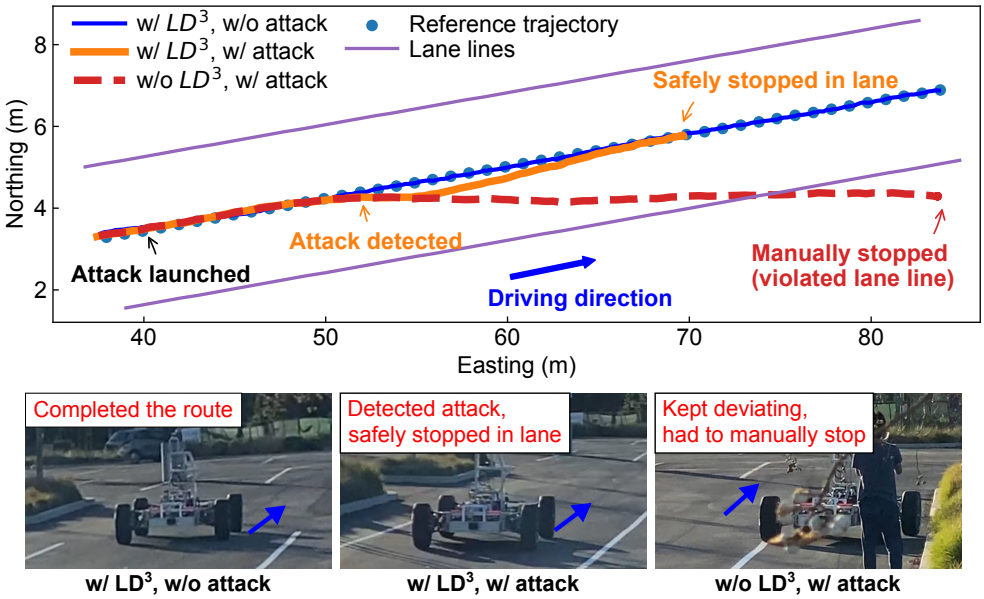

Figure 5.1: Physical-world end-to-end demonstrations of LD^3 using a Level-4 AD development chassis of real vehicle size and with full closed-loop control. (Top) Vehicle driving trajectories in bird's eye view. (Bottom) Final stopping positions under the three experimental settings. The driving direction and vehicle heading are annotated with blue arrows.

for attack detection, a tradeoff is that at which information level (i.e., GPS or MSF output) should we perform the detection. Recognizing that GPS outputs naturally have large noises and existing attack cannot deterministically predict when and where will large deviations occur in MSF, we decide to detect at the MSF output level to take advantage of such attack non-determinism. In the attack response (AR) stage, we choose to safely stop the vehicle in the ego lane, since this can minimize the attackable duration after detection and thus fundamentally bounds the attack-achievable deviation in the AR period. To account for the inherent LD-side adaptive attack surface introduced by LD^3, we further design a novel safety-driven fusion between LD and MSF that systematically penalizes the source that is more aggressive in causing lateral deviations, which can fundamentally reduce the attacker's capability in causing safety damages in AR period even in adaptive settings.

We evaluate our defense against the latest lateral-direction localization attack on a diverse set of real-world sensor traces with various environmental conditions. Our results show

that LD^3 is much more effective at detecting the attack compared to direct adaptation of physical-invariant based detection for small robotic vehicles [313]. Specifically, LD^3 can achieve effective detection with 100% true positive rates and 0% false positive rates on the sensor traces and the detections are timely when the lateral deviations are not yet large enough to touch the lane boundaries. Moreover, LD^3 is also effective at keeping the AD vehicle within the lane during the attack response periods, where the vehicle's final stopping deviations are *always* smaller than the lane straddling deviation. We also collect a night-time driving trace and find that LD^3 also has high defense robustness in low visibility conditions.

To further evaluate the defense in end-to-end driving with closed-loop control, we implement LD^3 on two open-source high-level AD systems, Baidu Apollo [7] and Autoware [213], and evaluate in an industrial-grade AD simulator [238] and the physical world with a real vehicle-sized AD chassis. Our results show consistent results in end-to-end drivings as in the trace-based evaluations. Fig. 5.1 shows the vehicle driving trajectories and stopping positions in the physical world experiments. As shown, LD^3 can promptly detect the attack and safely stop the vehicle at the center of the lane, while without LD^3, the vehicle drives out of lane boundary, and we have to manually stop the vehicle to prevent the collision. The demo videos of the simulation and physical world experiments are available at `https://sites.google.com/view/ld3-defense`.

Lastly, we explore two potential adaptive attacks against LD^3: (1) an ideal stealthy attack with the full knowledge of the defense aiming to evade the detection, and (2) the latest LD-side attack [327] against production AD systems. Results show that LD^3 can effectively bound the deviations of the stealthy attack from reaching the attack goals and can safely stop vehicle under the LD-side attack.

In summary, this work makes the following contributions:

- We perform the first systematic exploration of using LD to defend against lateral-

direction localization attacks on high-level AD systems. We quantitatively show that LD can provide comprehensive defense coverage for existing attacks despite the reliance on lane line markings, and is independent of the AD localization inputs.

- We design LD^3, a real-time defense solution including both attack detection and response stages. Evaluation on real-world sensor traces shows that LD^3 can achieve effective and timely attack detection, and can effectively stop the vehicle safely within the current lane. We also validate the robustness of LD^3 under low visibility conditions on a night-time driving trace.

- We implement LD^3 on two popular open-source AD systems, Baidu Apollo and Autoware, and evaluate the defense in end-to-end drivings in both simulation and the physical world.

- We evaluate LD^3 against two adaptive attacks and show that it is effective at bounding the deviations in the stealthy attack from reaching the attack goals and is robust to recent LD-side attack.

5.2 Threat Model

Attacker's capability. In this work, we assume the attacker can launch practical lateral-direction localization attacks through external means such as GPS spoofing, which can cause lateral deviations in the localization outputs. Specifically, we focus on the lateral-direction attacks since such attacks (1) can cause the AD vehicle to violate the traffic norm that a vehicle should be driving within its designated lane boundaries and should not have unexpected lane straddling behaviors, and (2) pose a direct threat to the AD vehicle and road safety, e.g., it can cause the AD vehicle to drive off highway cliff or onto the wrong way and being hit by other vehicles that failed to yield in time.

In particular, we do not consider simultaneous attacks that target both AD localization and lane detection at the same time, since such simultaneous attack neither already exists, nor can be easily achieved today (detailed discussions in §5.8).

AD control assumption. Same as *FusionRipper* [335] and also as a common design in academia [295] and industry [7, 213], we assume the AD systems are designed to drive at the center of traffic lane and constantly correct the deviations to the center. Since AD controllers constantly correct such deviations at a high frequency, e.g., 100 Hz [7], the lateral deviations in the AD localization will thus be directly reflected as physical world deviations, but to the opposite direction.

5.3 Lane Detection for High-Level AD Localization Defense

Motivation and novelty. Currently, no software-based defense solutions have been proposed to address the latest GPS spoofing-based lateral-direction localization attack in high-level AD systems. The closest ones are the recent physical-invariants based detectors proposed for small robotic vehicles such as drones and rovers, e.g., SAVIOR [313] and CI [126], which estimate the physical dynamics of drones and rovers to validate the GPS signal. Although they show high effectiveness for such small robotic vehicles under large attack deviation goals, their effectiveness in AD vehicle context is fundamentally more limited since (1) existing vehicle dynamics models have difficulties in modelling high-speed and curvy-road settings [225, 307]; and (2) in the AD context, the attack deviation goals can be much smaller (thus harder to detect) while still being safety-critical. As we concretely evaluate later in §5.5.2, direct adaptation of such existing physical-invariant based approach to the AD context suffers from very high false positives and is actually close to random guessing.

In comparison to small robotic vehicles, the AD context may also have its unique defense opportunities for such lateral-direction localization attacks. *Lane Detection (LD)* [186, 297], a technology commonly used in low-level AD systems for lane centering [131, 359], is such an example that can be used to measure the vehicle's lateral position within the current lane in real time, which is directly related to the lateral-direction attack goal (lane departure). Although effective in low-level AD systems (e.g., Level-2 ones such as Tesla Autopilot [359] that still count on human drivers to take over anytime), *LD is currently not used for high-level AD localization purpose (e.g., Level-4 ones such as Waymo that do not assume onboard human drivers)*. This is because what LD can provide is by nature only *local* positioning (i.e., relative positioning within ego lane), while high-level AD requires *global* positioning (i.e., in world coordinates on a map) for safe and correct driving decision-making without human drivers. Although there exist camera-based global localization methods using lane markings [212, 156], they are not generally adopted in state-of-the-art high-level AD localization [386, 170, 340, 29, 28, 31] as they are far from reaching the required centimeter-level accuracy [236, 319, 155].

While less suitable for global localization accuracy purposes in high-level AD, in this work we propose to be the first to explore novel use of LD for *defense purposes* in high-level AD localization. To concretely understand the potential of such a domain-specific defense opportunity, we analyze LD's defense properties in the following 5 general aspects.

1) General to lateral-direction localization attack. As mentioned above, LD can provide real-time information directly related to the *attack goal* of lateral-direction localization attacks. Thus, LD by nature has the potential to provide general defense capabilities to not only the existing attack designs such as FusionRipper [335], but also their potential adaptive versions or other new attack designs in the future, as long as the attack goal is to cause lateral deviations.

2) Technology maturity. Benefit from the growing prosperity of Deep Neural Networks

(DNNs), LD is already a mature technology that has been used for lane centering in low-level AD systems and vehicles, e.g., OpenPilot [131], Tesla Autopilot [359], GM Cadillac, Honda Accord, Toyota RAV4, Volvo XC90, etc. In fact, the existing camera-based LD solutions are quite robust to the dynamic environmental conditions. For example, Tesla Autopilot can effectively recognize lane lines even during a night storm [27]. Apart from DNN advancement, the camera auto-exposure and vehicle headlights also improve the usability of LD. Later in §5.5, we also evaluate our defense on datasets with various environmental conditions and show that it is robust to low visibility conditions.

3) Defense deployability. Since today's high-level AD vehicles are all equipped with cameras for road object detection, using them for an LD-based defense solution is thus readily deployable without the need to install any new hardware. Moreover, many state-of-the-art LD models are publicly available [297, 286], including those used in industry-grade lane centering systems [131]; some high-level AD systems are also using LD for camera calibrations [7].

4) Defense coverage. For LD to be effective, the road at least needs to have lane line markings, which may not be available in local road segments such as intersections. Interestingly, due to real-world sensor noises and algorithm inaccuracies, the attacks to MSF localization are *fundamentally opportunistic*. For example, despite having a high overall attack success rate, the latest lateral-direction localization attack cannot predict when and where a large deviation can be injected to the MSF outputs [335]. Due to such opportunistic property, the attacker *cannot deterministically cause a desired lateral deviation to appear and only appear in regions without lane line markings*. Such an attack property is fundamental to the MSF localization designs popularly used in high-level AD systems, since with such a design the attack effectiveness is fundamentally dependent on sensor noises and algorithm inaccuracies of other sources, which are neither observable nor controllable by a tailgating attacker [335].

Motivated by this insight, we analyze all attack traces evaluated in the *FusionRipper* pa-

per [335] and our own evaluation later (§5.5.1), and find that LD can indeed provide a decent practical defense coverage: among all attack starting points in the traces, only *0.8% (15/1813)* achieved the attack goal in road regions without lane line markings. Thus, an LD-based defense, if effective, can already provide protection for the 99.2% of the possible attack attempts. In addition, autonomous trucks, which are an important high-level AD application, are generally not subject to such limitation since they mainly operate on the "middle mile" (i.e., highways) [38, 46], where lane line markings are generally always available.

5) Independence to existing localization attack. To defend against existing attacks, a desired defense property is that the lane line markings perceived by LD are not already used in MSF localization. This is because if such information is already used, existing attacks might have already exploited their vulnerable periods (e.g., natural detection inaccuracies), making the additional use of such information for defense less likely to be effective. In representative MSF localization designs, the LiDAR locator is the only one among the MSF inputs that is possible to utilize lane line markings as features. Thus, we perform an experimental analysis to understand the dependency between state-of-the-art LiDAR locators [386, 213] and lane line markings in Appendix F. Our results show that today's LiDAR localization algorithms have a *statistically-strong independence* of the lane line markings, very likely because lane markings are much less useful for global localization on a map compared to more unique road features such as buildings, roadside layouts, and traffic signs. This thus suggests that LD can indeed provide independent defense information to existing attacks. However, such independence property will disappear in adaptive attack settings (i.e., consider attacking LD after the defense is deployed). Thus, we require our defense design to be fully-aware of such adaptive attack surface (§5.4.2), and also evaluate it later (§5.7).

5.4 Novel LD-based Defense Design: LD^3

Considering the multi-dimensional defense opportunities above, in this work we are motivated to design the first domain-specific lane detection-based defense approach against lateral-direction AD localization attack, named LD^3 *(Lane Detection based Lateral-Direction Localization attack Defense)*. In this section, we first describe the associated design challenges and then present the design details.

5.4.1 Design Challenges

Although LD comes with various defense opportunities, systematically leveraging it for AD localization defense purpose still needs to address the following main design challenges:

C1: Non-trivial design details for attack detection. Although at the high level LD can provide information directly related to lateral-direction attacks (i.e., lateral deviation to lane departure), at the detailed defense design level there are still many technical challenges we need to address, for example (1) incompatibility of the coordinate systems, i.e., LD is by default in *local* positioning coordinate system (i.e., within the ego lane), while the attack is in *global* coordinate system (i.e., the world coordinates); (2) choice of the attack-influenced information level for attack detection, e.g., directly at the spoofed GPS signal level or at the attack-influenced MSF output level; and (3) sufficient robustness to natural LD inaccuracies in practice, e.g., missing or incorrect detection, for minimizing possible false positives in attack detection.

C2: Need for AD-specific attack response design. Since high-level AD vehicles are travelling at high speed and by design cannot assume on-board human driver ready for take-over at any time (already the case in some commercial AD services [222, 87]), it is necessary to further design an attack response step that can (1) minimize the safety risks during response,

and (2) assume no dependence on human assistance. For small robotic vehicles such as drones and rovers, prior works have considered using state estimation models to replace the attacked physical sensor after attack detection [125, 416]. However, such methods still count on human operators to take over as soon as possible since such state estimations cannot replace physical sensors for a prolonged duration due to drifting [125], not to mention that such models are suffering from much more severe motion model accuracy limitations when applied to the AD context (§5.5.2). Thus, a new design is needed to achieve our AD-specific response goal above.

C3: Adaptive attack from LD side. While LD is currently independent to existing high-level AD localization attacks due to the lack of use (§5.3), our defense-purpose use of it in LD^3 is inherently introducing a new attack surface from the LD side. In fact, recent works have already discovered concrete lateral-direction attacks against LD in production AD context [327]. To systematically account for such inherent adaptive attack surface, our defense design thus needs to consider the more challenging setup where both the attack detection and response designs cannot simply assume the LD side is trustworthy (and use it as the benign reference accordingly) when its outputs are inconsistent with the AD localization side.

5.4.2 Design Overview

In this section, we explain each design component in LD^3 and how they address the above design challenges. Fig. 5.2 shows an overview of LD^3 fitted in a typical high-level AD system.

Attack detection at MSF output level. As shown in Fig. 5.2, the attack detection step is performed in the localization module to constantly check the consistency between the LD outputs and original localization output and raise anomalies use popular anomaly detectors such as CUmulative SUM (CUSUM). To address the incompatibility of their coordinate

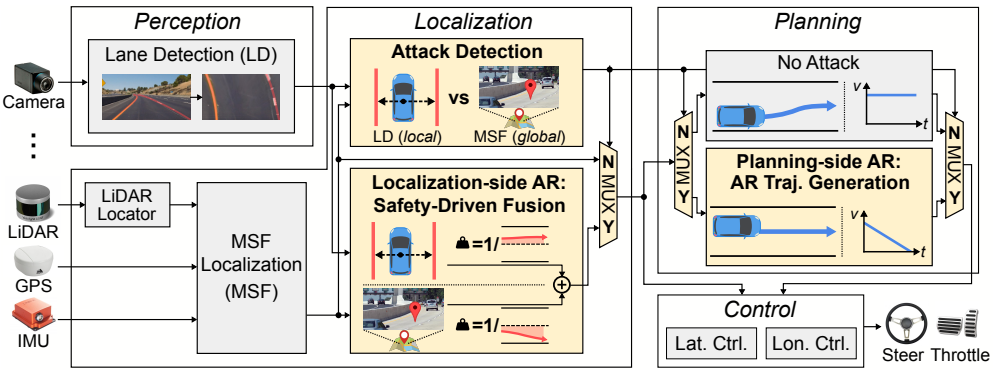

Figure 5.2: Overview of LD^3 design integrated in a typical high-level AD system. New components are highlighted in yellow.

systems mentioned in *C1*, we convert both into a unified lateral deviation representation w.r.t. the *lane centerline* since that's directly related to the lateral-direction attack goal. Regarding the choice of the attack-influenced information level for attack detection, we choose to detect at the MSF output level rather than at the GPS output level since (1) in normal conditions, GPS positions can naturally have large noises while MSF outputs are at centimeter-level accuracy [386]. Thus, performing the detection at the MSF level can better reduce false positives; and (2) detecting at the MSF output level also allows LD^3 taking advantage of the *opportunistic* property of *FusionRipper*, for which the attacker cannot predict where and when will MSF exhibit large deviations. This thus can make it much more difficult for the attacker to easily bypass the detection by targeting locations without lane line markings. We also have designs for addressing false positives from common lane detection inaccuracies.

Attack response via safe in-lane stopping. As discussed in *C2*, we need a new AD-specific design for the Attack Response (AR) step. There are several common choices in human driving if the vehicle navigation is malfunctioning, for example maintaining driving in the current lane waiting for the system to recover, or pulling over to the roadside. However, these cannot apply to the context of AD localization attacks, since without knowing the accurate real-time location, we cannot even know how to safely and correctly drive in the

current lane or to the roadside. We also cannot blindly count on the LD outputs to drive due to the need to account for the adaptive attack surface on the LD side (*C3*). Thus, we consider the safest AR choice is to try to safely stop in the current lane, which has the minimal reliance on the attack-time localization accuracy for maximizing safety in the AR period. More importantly, on the attack side, since this minimizes the attackable duration after detection, it can fundamentally *bound* the attack-achievable deviation in AR period.

Safety-driven fusion for adaptive LD attack. Although the in-lane stopping AR strategy can already bound the attack-achievable deviation, it is still highly desired if we can minimize the attacker's impact on the localization accuracy during the AR period, since to safely stop, there is still a long stopping distance that the ego vehicle has to travel, especially when the speed is high (e.g., over 50 meters at 60 mph [284]). To account for the adaptive LD attack surface (*C3*), the key challenge is how to decide which side (LD or MSF) to trust when they are conflicting with each other in AR. Motivated by the safety-first principle in production AD design [85], we propose a novel *safety-driven fusion* design, which systematically decides the contributions from different fusion inputs based on their tendencies to cause unsafe driving; the higher such tendency is, the smaller their contributions will be to the final fusion output. In our problem context, such a tendency is judged by the deviation aggressiveness to cause lane departure, which will thus by default penalize the attacked side no matter it is from LD or MSF, leading to less attack-introduced deviation. To bypass this penalty and more effectively influence the fused results, the attacked side has to be less aggressive in lateral deviations. However, given the limit on the attackable duration imposed by the in-lane stopping AR strategy, the attack-achievable deviation during AR will still be reduced. Thus, under our AR design that bounds the attackable duration, such safety-driven fusion design can further fundamentally reduce the attacker's capability in causing safety damages during AR even in adaptive settings.

Algorithm 2 Attack detection by checking the consistency between MSF and LD

Notations: MSF: MSF position output; LD: lane detection output; S, b, τ: CUSUM statistic, weight, anomaly threshold; D: deviation to lane centerline; lw_{map}: lane width from semantic map

Initialize: $S_0 \leftarrow 0$

for each new lane detection output LD_i **do** ▷ e.g., runs at 20Hz
  $MSF_i \leftarrow$ latest MSF position ▷ MSF is often more frequent than LD
  $D_i^{\text{MSF}} \leftarrow \text{MapLaneDev}(MSF_i)$ ▷ MSF dev. (Table 5.1)
  $lw_{\text{map}} \leftarrow \text{MapLaneWidth}(MSF_i)$ ▷ lane width (Table 5.1)
  $D_i^{\text{LD}} \leftarrow \text{LdDev}(LD_i, lw_{\text{map}})$ ▷ LD dev. to centerline (Alg. 5)
  $S_i \leftarrow \max(0, S_{i-1} + |D_i^{\text{MSF}} - D_i^{\text{LD}}| - b)$ ▷ calc. CUSUM statistic
  if $S_i > \tau$ **then**
    attacked $\leftarrow$ true ▷ report under attack if over threshold
    break
  end if
end for
$\Rightarrow$ *switch to attack response*

Table 5.1: Semantic map APIs required for LD[3].

Map API	Description
`MapLaneDev(pose)`	Query the deviation from `pose` to the closest lane centerline
`MapLaneWidth(pose)`	Query the width of the closest lane to `pose`
`MapLanePoint(pose)`	Query the closest point and lane heading on the closest lane centerline to `pose`
`MapIsIntersection(pose)`	Query if `pose` is located in an intersection

5.4.3 Attack Detection Design

As described above, we choose to perform the attack detection at the MSF output level, which is thus designed as a post-processing step in the localization module as shown in Fig. 5.2. The detection algorithm is shown in Alg. 2. As mentioned in *C1*, the MSF and LD outputs are in different coordinate systems. Therefore, we first need to convert them to a unified coordinate system such that they are comparable. For MSF outputs, we obtain an *MSF-based lateral deviation to the lane centerline* (D_i^{MSF} in Alg. 2) by querying the MSF position in the semantic map [260], which is a standard utility on high-level AD systems storing the road geometry information of the area that the AD vehicle is allowed to drive. For the LD outputs, we can calculate the lateral deviation to the centerline based on the left and right lane line polynomial functions (detailed in Appendix E). However, real-world lane markings can be complicated and confusion sometimes. For example, it is common to

find that one of the lane lines missing or incorrectly detected in regions with lane splitting and merging. Therefore, we design two optimizations to calculate a more robust lateral deviation from the LD outputs leveraging the lane width from the semantic map (detailed in Appendix E), which is a problem-specific improvement opportunity since in the main LD usage domain, low-level AD systems, such semantic maps are not generally available. Since LD[3] relies on the existence of lane line markings, we disable the attack detection prior to entering these regions based on the information from the semantic map.

After obtaining the MSF- and LD-based lateral deviations, we can then use their deviation consistency to determine if MSF localization is under attack. To do so, we apply the widely-used CUSUM anomaly detector (line 6–10 in Alg. 2), which has shown high detection effectiveness in prior works [377, 313]. The CUSUM detector calculates a statistic $S_i = max(0, S_{i-1} + |r_i| - b); S_0 = 0$, where $r_i = D_i^{\text{MSF}} - D_i^{\text{LD}}$ is the residual between the MSF and LD lateral deviations, b is a weight to prevent the CUSUM statistic from monotonically increasing in the benign scenarios. We consider as under attack if S_i is over a certain threshold τ. Once an attack is detected, we then switch to the Attack Response stage.

5.4.4 Attack Response Design

As described in §5.2, we consider safe in-lane stopping as the safest AR choice. As shown in Fig. 5.2, the AR is composed of two components to safely drive the vehicle before stop: (1) AR trajectory generation on the planning side, and (2) safety-driven fusion of MSF and LD on localization side.

Planning-side AR: AR trajectory generation. The planning module in the high-level AD system periodically generates planned trajectories, for which the controllers take as speed and lateral position reference to produce throttling and steering commands. Thus, to enforce the AR goal, the planning module needs to generate an AR trajectory with a

Algorithm 3 Safety-driven fusion for attack response

Notations: D: deviation to lane centerline; P: uncertainty from MSF or LD outputs; MSF: MSF position output; kf: 1-dimensional Kalman Filter; R: uncertainty for KF update

function FUSEDPOSE(D^{MSF}, D^{LD}, P^{MSF}, P^{LD}, MSF)
  $R^{\text{MSF}}, R^{\text{LD}} \leftarrow$ UNCERTAINTY(D^{MSF}, D^{LD}, P^{MSF}, P^{LD})
  $kf.update(D^{\text{MSF}}, R^{\text{MSF}})$; $d \leftarrow kf.predict()$
  $kf.update(D^{\text{LD}}, R^{\text{LD}})$; $d \leftarrow kf.predict()$
  $pose_{\text{center}}$, $heading_{\text{center}} \leftarrow$ MAPLANEPOINT(MSF) ▷ Table 5.1
  $pose_{\text{fusion}} \leftarrow$ ADDDEVTOPOINT($pose_{\text{center}}$, $heading_{\text{center}}, d$)
  return $pose_{\text{fusion}}$
end function

stopping motion. Since our AR goal is to stop in the ego lane, we design the AR trajectory to be aligned with the lane centerline. To reduce the speed, we then set a slowing-down speed profile on the AR trajectory based on a safe deceleration value used in high-level AD systems (Appendix E). Note that since the original planning algorithms are typically designed under the assumption that the localization accuracy is high (i.e., cm-level [319]), we find directly re-using such algorithms in AR will result in unstable control since the planned trajectories are too sensitive to the larger localization errors and uncertainties after fusing the LD and MSF sides when one side is under attack. Thus, we directly set the planned trajectory as the centerline of the ego lane to achieve more stable control.

Localization-side AR: safety-driven fusion. As described in §5.4.2, we need to design a safety-driven fusion algorithm on the localization side that can systematically fuse LD and MSF outputs while taking less contributions from the side that is more aggressive in causing lateral deviations. To achieve this, we leverage a classic fusion algorithm design, *Kalman Filter* (KF) based fusion, which can systematically determine the contributions of each fusion source using *uncertainties* [361, 166]. In the original design, the uncertainty score calculation are based on the noise-level measurements reported by the sources themselves, which thus are not suitable in attack settings since such measurements are also fundamentally under the attacker's control.

To systematically realize our safety-driven fusion design between LD and MSF, we thus

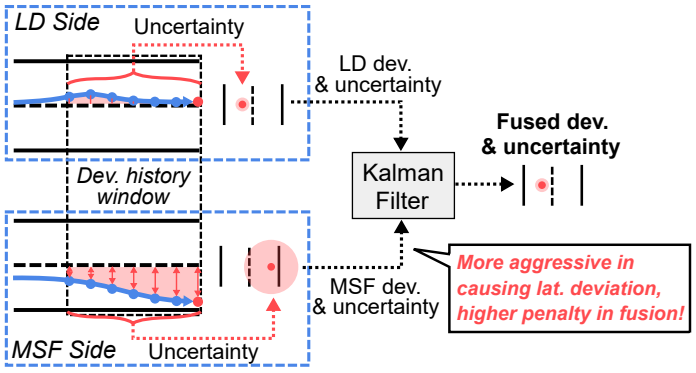

Figure 5.3: Illustration of safety-driven fusion in the attack response.

still leverage such uncertainties-based fusion framework but design novel uncertainty score calculation based on their tendencies to cause lane departure. Alg. 4 lists the pseudocode for the uncertainty calculation. As shown (line 2–7), we store the historical lateral deviations from MSF and LD in two fixed-size windows. To obtain the uncertainties, we first calculate the cumulative deviations in these two windows, and then calculate their proportions to the geometric mean of them (line 8–11). We choose geometric mean over arithmetic mean since it can better penalize the source with a larger cumulative deviation. To increase the design flexibility, we include both our cumulative lateral deviation based uncertainty and the uncertainty from MSF/LD algorithms in the final uncertainty and use a weight λ to adjust their fractions (line 12–13).

With the uncertainties, we apply standard KF update/predict operations to fuse the MSF and LD lateral deviations (line 3–4 in Alg. 3). We then add the fused lateral deviation to the closest centerline point along the lateral direction based on the lane heading to instantiate a fused localization in the global coordinate system (line 5–6 in Alg. 3). Fig. 5.3 illustrates an example of the safety-driven fusion process.

Algorithm 4 Cumulative lateral deviations based uncertainties calculation

Notations: D: deviation to lane centerline; P: uncertainty from MSF or LD outputs; DS: deviation history; w: deviation history window size; λ: weight of the deviation history based uncertainty

Initialize: $DS^{\text{MSF}} \leftarrow \{\}$; $DS^{\text{LD}} \leftarrow \{\}$

function UNCERTAINTY(D^{MSF}, D^{LD}, P^{MSF}, P^{LD})
  $DS^{\text{MSF}} \leftarrow DS^{\text{MSF}} \mid\mid |D^{\text{MSF}}|$ ▷ append MSF dev. to the history
  $DS^{\text{LD}} \leftarrow DS^{\text{LD}} \mid\mid |D^{\text{LD}}|$ ▷ append LD dev. to the history
  if size of $DS^{\text{MSF}} > w$ **then** ▷ remove first element if full
    $DS^{\text{MSF}} \leftarrow DS^{\text{MSF}} \setminus DS^{\text{MSF}}[0]$
    $DS^{\text{LD}} \leftarrow DS^{\text{LD}} \setminus DS^{\text{LD}}[0]$
  end if
  $s^{\text{MSF}} \leftarrow \sum_{n=1}^{w} DS^{\text{MSF}}$; $s^{\text{LD}} \leftarrow \sum_{n=1}^{w} DS^{\text{LD}}$ ▷ dev. sums
  $s^{GeoMean} \leftarrow \sqrt{s^{\text{MSF}} \cdot s^{\text{LD}}}$ ▷ geometric mean of dev. sums
  $f^{\text{MSF}} \leftarrow s^{\text{MSF}} / s^{GeoMean}$ ▷ dev. history based uncertainty for MSF
  $f^{\text{LD}} \leftarrow s^{\text{LD}} / s^{GeoMean}$ ▷ dev. history based uncertainty for LD
  $R^{\text{MSF}} \leftarrow \lambda f^{\text{MSF}} + (1-\lambda) P^{\text{MSF}}$ ▷ MSF uncertainty
  $R^{\text{LD}} \leftarrow \lambda f^{\text{LD}} + (1-\lambda) P^{\text{LD}}$ ▷ LD uncertainty
  return $R^{\text{MSF}}, R^{\text{LD}}$
end function

5.5 Defense Effectiveness Evaluation

In this section, we evaluate LD^3 against the state-of-the-art lateral-direction attack targeting high-level AD localization.

5.5.1 Evaluation Methodology

Targeted AD system and attack. Since LD^3 is designed for high-level AD systems, we choose the industry-grade full-stack Baidu Apollo AD system [7] as a representative prototyping target. Specifically, Baidu Apollo adopts an MSF-based localization highly representative in both design (KF-based MSF) and implementation (state-of-the-art localization accuracy [386]). Note that although our evaluation here uses Baidu Apollo, the LD^3 design itself is generalizable to other industry-grade high-level AD systems; for example, later in §5.6.2 we also implemented it in Autoware for end-to-end physical evaluation. For targeted attacks, we evaluate against the recent *FusionRipper* attack [335] since it is (1) the state-of-the-art and only lateral-direction localization attack that can break MSF localization; and

(2) directly applicable to the above representative MSF implementation.

Real-world sensor traces and *FusionRipper* attack effectiveness. Since our evaluation target is the *FusionRipper* attack, we follow the same evaluation methodology as in [335] and conduct our evaluation on real-world sensor traces from the KAIST complex urban dataset [203]. Specifically, we look for traces with camera data as required by LD^3, select the ones that the Apollo MSF can stably operate without attack [335], and apply *FusionRipper* from each consecutive timestamp as in [335]. In total, we obtain 562 attack traces summarized in Table 5.2. These traces cover diverse driving scenarios, e.g., different road types (344 on local roads and 218 on highways), driving speeds (9.5 to 26.3 m/s), time-of-day (e.g., 36 in the morning, 182 around sunset time), and road conditions (e.g., 170 with snow on road).

For each trace, we follow the same method to identify the most effective attack parameters as in the *FusionRipper* paper. Note that in the table our attack goal deviation is larger than the *FusionRipper* paper since they focus on the minimum urban lane width (i.e., 2.7 m) while we set the attack goal in a more realistic setting by measuring the lane widths in the dataset. This does not affect the attack effectiveness; as shown, the overall attack success rate is over 98%, which is consistent with the *FusionRipper* paper. In our evaluation, we exclude the scenarios without lane markings (e.g., when the vehicle is in an intersection) since it is out of the applicable domain for LD^3. As analyzed in §5.3, the lack of coverage of such scenarios do not eliminate the defense value since only 0.8% of the attacks can possibly succeed in such scenarios and such successes are out of the attacker's control.

Lane detection and AD control effects under attack. Since LD^3 does not assume any specific requirement on the lane detector, we are free to use any state-of-the-art lane detector or even an ensemble of lane detectors. In our evaluation, we opt to the LD model used in OpenPilot [131], which is already used commercially for Automated Lane Centering. The KAIST traces include time-synchronized left and right camera frames from a front-

Table 5.2: Details of the 562 total attack traces used in our evaluation and the *FusionRipper* attack effectiveness.

	Attack Trace #	Road Type	Avg. Speed	FusionRipper Attack			
				Attack Goal Dev	Best d	Best f	Success Rate
ka-local31	174	Local	10.9m/s	1.3m	0.5	1.2	99.4%
ka-local33	170	Local	9.5m/s	1.3m	0.3	1.3	98.3%
ka-highway36	182	Highway	26.3m/s	1.9m	0.3	1.3	100%
ka-highway18	36	Highway	24.8m/s	1.9m	0.3	1.3	100%

facing stereo camera. In our evaluation, we regard the left and right cameras as independent cameras and run the LD model and calculate lateral deviations (Alg. 5) separately on them. We then aggregate their results to obtain an averaged lateral deviation on the LD side.

Since KAIST traces are collected under benign driving, we need to model the LD outputs when the AD localization is under attack. Same as the *FusionRipper* paper [335], we assume the lateral deviations in the MSF localization will be directly reflected as physical world deviations to the *opposite* direction (§5.2). We then model the attack-influenced LD outputs by adding the physical world deviations to the lateral deviations calculated from the benign LD outputs. Later in §5.6, we evaluate LD^3 in both end-to-end simulation and physical-world environments without such an assumption.

Baseline: SAVIOR. As a baseline, we evaluate the attack detection effectiveness of the closest alternative software-based method based on latest prior works for small robotics vehicles such as drones and rovers: physical-invariant based defenses [313, 126]. Specifically, we select SAVIOR [313] as a representative design since it adopts more principled state estimation models and thus shows superior detection performance over prior designs such as CI [126]. The detailed setup for SAVIOR evaluation can be found in Appendix G.

Evaluation metrics. As LD^3 involves two defense stages with different defense goals, we separate the evaluation into attack detection and response evaluations. For attack detection evaluate, we plot the *ROC curves* to systematically show the TPRs and FPRs under dif-

ferent CUSUM parameters b and τ (§5.4.3). In addition to ROC curves, we also report the maximum MSF lateral deviation before the attack is detected by LD^3. This *detection deviation* is a metric to indicate the detection timeliness, e.g., a detection deviation smaller than the lane straddling deviation (i.e., deviation to touch the lane line) means that the attack is detected early in time before it can cause any meaningful adversarial consequences. For attack response evaluation, we focus on the lateral deviations since our AR goal is to steer the vehicle to stop within the lane boundaries. In particular, we report two lateral deviation metrics with one measuring the *maximum deviation* before the vehicle fully stops, and another one measuring the final *stopping deviation*. In practice, the latter is more important since it will be the permanent deviation after the vehicle stops.

5.5.2 Attack Detection Effectiveness

Attack detection rates. The top figures in Fig. 5.4 show the detection ROC curves of LD^3 against *FusionRipper*. As shown, LD^3 can achieve effective detection with 100% TPRs and 0% FPRs on all 4 traces. During the searching for best CUSUM parameters, we find that in benign drivings, the differences between MSF and LD lateral deviations are always bounded within certain range (<0.6 m). However, in attacked drivings, *FusionRipper* will cause larger lateral deviations on the MSF side, which will also be reflected on the LD side in the opposite direction. Such a difference between benign and attacked drivings makes the attack easily detectable by LD^3. Fig. 5.5 shows an example of the benign and attacked MSF and LD lateral deviations and their CUSUM statistics. In the attacked case, *FusionRipper* launches the vulnerability profiling stage from t=1544686730 and discovers a vulnerable window at t=1544686765, where MSF starts to exhibit larger lateral deviations. Because of the distinctive MSF/LD consistency levels between the benign and attack cases, it is thus straightforward to set a CUSUM threshold to differentiate them.

Baseline comparison. As shown, SAVIOR's detection performance is only slightly better than random guessing and far from being an ideal detector. Such a poor detection performance would render SAVIOR unpractical since it will introduce lots of false positives in normal driving.

The reason behind the poor detection performance in the AD context is twofold. First, compared to drones and rovers, the physical dynamics of the vehicle are much harder to model due to the complex physical moving characteristics, e.g., tire-road frictions, aerodynamic forces, road bank angles, etc. [315]. For example, prior study [307] finds that the error of kinematic bicycle model increases very fast at high speeds (e.g., 25 m/s) or on curvy roads (e.g., steering angle at 4°). In comparison, the bicycle model used in SAVIOR is reported having an average position error of 0.33 m *within 0.8 sec* under low-speed settings (e.g., 13.8 m/s) in [225] and its error keeps accumulating as time progresses; comparably, the same bicycle model incurs an average error of 1.076 m on *ka-local31* within 1 sec, where the trace contains many turns and curvy roads.

Second, the attack deviation goals in the AD context can be much smaller but still being safety-critical. While SAVIOR is effective at detecting attacks on small robotics vehicles such as drones with large deviation goals (e.g., ∼50 m [313]), attacks targeting high-level AD systems requires much smaller deviation and thus harder to detect. For example, even lateral deviations <0.5 m are enough to cause lane departure on narrow urban roads (e.g., 2.7 m wide [344]).

Attack detection deviations. To evaluate attack detection deviations, we choose a CUSUM weight $b = 0.6$ and threshold $\tau = 0.1$, which can achieve best attack detection effectiveness on all traces. For example, the detection deviation in Fig. 5.5 is 0.36 m under these CUSUM parameters. The bottom figures in Fig. 5.4 show the distributions of maximum deviations *FusionRipper* has reached before being detected by LD^3 (box plots with pink background). As shown, LD^3 can promptly detect the attack before it can even cause

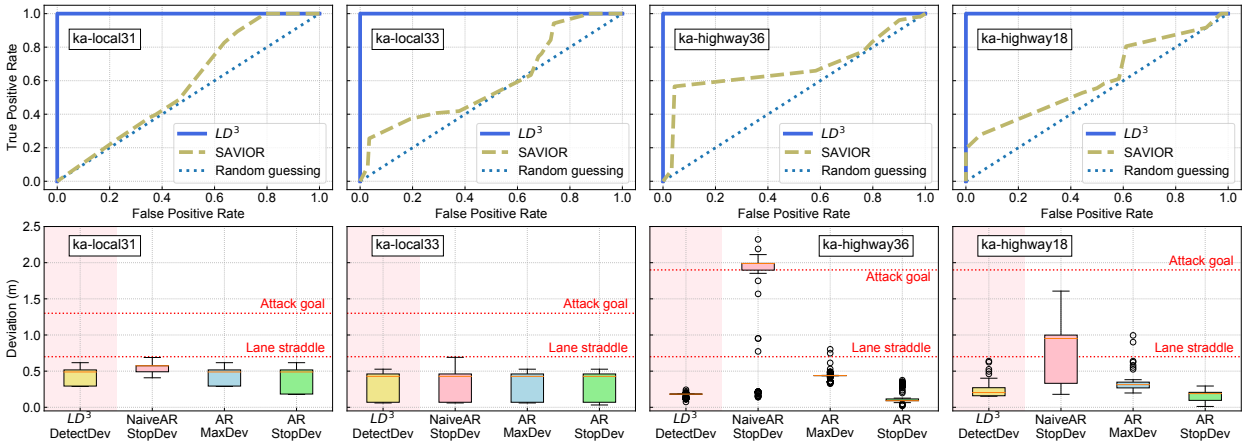

Figure 5.4: (Top) Attack detection ROC curves; (Bottom) Detection and Attack Response (AR) deviations in the LD^3 evaluation.

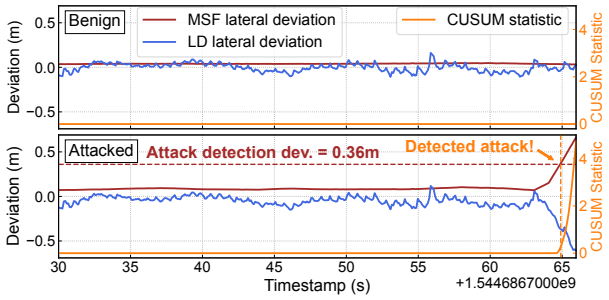

Figure 5.5: Benign and attacked MSF/LD lateral deviations and CUSUM statistics.

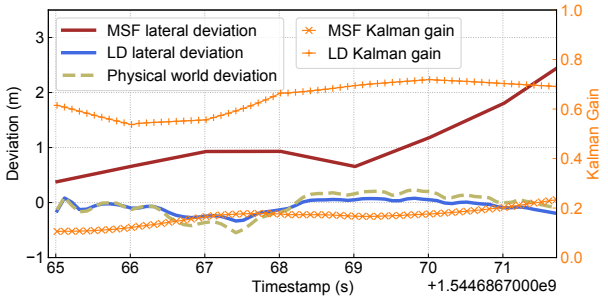

Figure 5.6: MSF/LD and physical world deviations and Kalman gains during AR period.

lane straddling, and the average detection deviations are all below 0.5 m. However, there do exist two attack cases in *ka-highway18* that the detection deviations are close to the lane straddling deviation. This is because in these attack cases, the lateral deviation at the MSF side raises very rapidly between detection intervals such that the deviation has already reached a large number (e.g., 0.63 m) before LD^3 has a chance to perform the detection. Nevertheless, none of the attack cases are detected after *FusionRipper* starts to cause lane straddling and all of them are far away from reaching the attack goal deviation.

5.5.3 Attack Response Effectiveness

The distributions of the maximum deviations and final stopping deviations are shown in the bottom figures in Fig. 5.4 (box plots without background colors). During the AR periods, none of the attack cases have a maximum or stopping deviation over the attack goal deviation (1.3 m for local and 1.9 m for highway). Despite 4 attack cases on the highway traces have maximum deviations exceed the lane straddling deviation (0.7 m), their stopping deviations are all corrected back to be within the lane boundaries. This shows that our AR design (§5.4.4) is effective at keeping the vehicle within the lane boundaries when it stops, which can prevent the much more dangerous situation where it stops out of the ego lane.

Moreover, comparing between local and highway, the highway traces often have *larger maximum deviations* and *smaller stopping deviations*. This is because the driving speeds when the attacks are detected on the highway traces (27.3 m/s on avg.) are much higher than that on the local traces (3.8 m/s on avg.). This leads to a much longer AR period on highways ($\sim$7 sec on avg.) than that on local roads ($<$1 sec on avg.). As a result, *FusionRipper* can keep causing larger lateral deviations after the attack is detected, but in the meanwhile, our AR design can also correct more given the longer AR period.

Fig. 5.6 shows an example of the MSF/LD and physical world deviations during the AR

period on *ka-highway36*, where the maximum deviation and stopping deviation are 0.52 m and 0.19 m, respectively. In this example, since *FusionRipper* keeps increasing the deviation on the MSF side, the safety-driven fusion (§5.4.4) penalizes the lateral deviations on the MSF side with higher uncertainties and thus results in smaller Kalman gains, which indicate the weights of the inputs in KF update. Consequently, the fusion process prioritizes the lateral deviations on the LD side, which are similar to the physical world deviations, and the lateral controller thus can steer the vehicle towards the right direction.

Comparison with naive AR design. A naive AR design, named NaiveAR, applies the maximum deceleration to stop but still keeps using the MSF outputs for steering. Such design is similar to the *in-lane stop* planning scenario that Baidu Apollo adopts to handle emergencies [93]. To evaluate this, we record the MSF lateral deviations at the end of AR periods and regard them as the stopping deviations based on the control assumption (§5.2). The stopping deviations of NaiveAR are shown in Fig. 5.4. Because of the longer AR periods on the highway, the stopping deviations under NaiveAR are significantly higher than that using our complete AR design, especially on the highway traces. In particular, since the lateral deviations on *ka-highway36* increase very quickly, over 75% of the attacked cases still reaches a lateral deviation higher than the attack goal deviation, which consequently leads to >75% attack success rate for *FusionRipper* on *ka-highway36* despite the attacks are correctly detected. On the other hand, with the complete AR design, none of the attack cases can be even deviate out of the lane boundaries.

5.5.4 Evaluation under Limited Visibility

Trace collection and defense evaluation setup. We collect a *night-time* driving trace at around 11 p.m. our local time using an Advanced Driver-Assistance System (ADAS) device named EON [130], which is the official device to run OpenPilot [131]. Specifically, we record

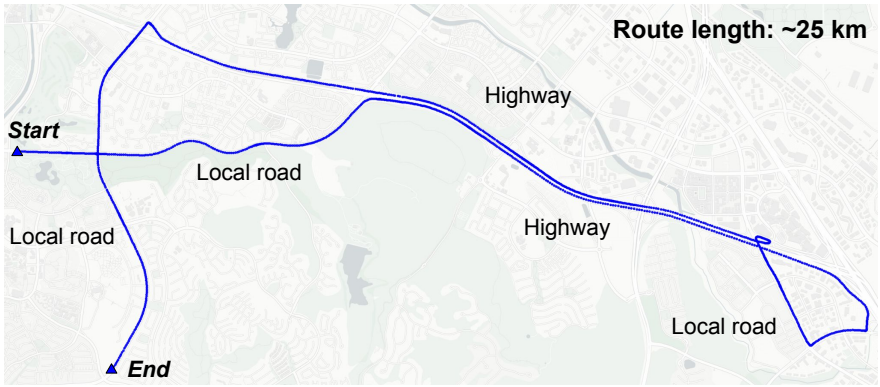

Figure 5.7: Route of the night-time sensor trace collected using EON [130].

the localization and LD outputs during the trace collection for the defense evaluation. The trace is ∼25 km in length with 3 local road and 2 highway segments as shown in Fig. 5.7. Since EON does not provide LiDAR data, we are not able to run MSF and *FusionRipper* attack. To model the attack effect, we apply the lateral deviations from the *most aggressive* attack trace in *ba-local* trace used in the *FusionRipper* paper [335] to the localization outputs, which only takes 10 sec from the start of attack to reaching a 2 m lateral deviation. This is similar to the prior works where they directly apply the attack traces in the target systems for attack detection evaluation [313, 126]. Specifically, we apply the attack trace consecutively to all road segments excluding the intersections, which results in 98 attacked and 98 corresponding benign segments in total.

Defense effectiveness. Similar to results on KAIST traces (§5.5.2), LD^3 can achieve effective attack detection with 100% TPR and 0% FPR on the night-time trace. The attack detection and AR deviations are shown in Fig. 5.8. As shown, even under such low-light condition, LD^3 can still timely detect the attack with an average detection deviation of 0.29 m. Consistent with findings in §5.5.3, the stopping deviation on the real vehicle trace is only 0.17 m on average, which means LD^3 is effective at stopping the vehicle within the lane boundaries. In comparison, NaiveAR has a stopping deviation much higher than LD^3, where one attack segment (maximum deviation is 1.33 m) exceeds the goal deviation for local roads.

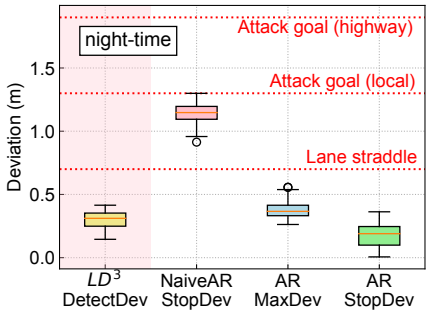

Figure 5.8: Detection and AR deviations on the night-time trace.

5.6 End-to-End Evaluations

In this section, we implement LD^3 on 2 open-source full-stack AD systems, Baidu Apollo [7] and Autoware [213], and evaluate LD^3 under end-to-end drivings in both simulation and the physical world. The demo videos are available on our project website at `https://sites.google.com/view/ld3-defense`.

5.6.1 Evaluation in AD Simulator

Experimental setup. We implement LD^3 in Baidu Apollo v5.0.0 [7] and evaluate under 4 driving scenarios with different driving speeds (local road and highway speeds) and road geometries (straight and curvy roads) in the LGSVL simulator [238]. In our evaluation, we include the LD^3 variant with naive AR design (§5.5.3) and one without defense. We repeat the simulation for 10 times with different attack starting times for each combination of simulation scenarios and defense settings.

In our evaluation, we reuse the SCNN model [297] in Baidu Apollo for LD, which is currently used only for camera calibration in Baidu Apollo. Specifically, we run the complete Baidu Apollo AD system with all functional modules enabled in an industry-grade AD simulator, LGSVL [238]. Since LGSVL does not provide LiDAR locator maps required for MSF, we

instead run Baidu Apollo localization in the Real-Time Kinematic mode, which directly takes the ground truth positions from LGSVL. To simulate the *FusionRipper* attack effect, we add the lateral deviations from the same attack trace used in §5.5.4 to the localization outputs.

To evaluate LD^3, we create 4 driving scenarios on two LGSVL maps: Single Lane Road (SLR) and San Francisco (SF). Specifically, the SLR map is a long straight road, and we create a low-speed (SLR-Low) and high-speed (SLR-High) driving scenario on it by adjusting the maximum cruising speed in Apollo planning. The SF map is a 1:1 re-creation of a portion of the San Francisco city, from which we select a straight (SF-Straight) and a curvy road (SF-Curvy).

Results and demos. Our simulation results show that the attack detection rates for both LD^3 and LD^3-NaiveAR are all 100% in the 10 runs, and none of the benign drivings are falsely detected as under attack. Table 5.3 shows the maximum lateral deviation achieved in the whole simulation (including both attack detection and response periods) in each scenario/defense setting and the corresponding vehicle stopping location. As shown, with LD^3, the average maximum deviations are smaller than lane straddling deviation in all 4 scenarios and the vehicle can always safely stop in the lane. In comparison, due to the blind trust of the localization outputs in the AR period, LD^3-NaiveAR has much higher maximum deviations than LD^3 and the vehicle's stopping locations are either lane straddling or already crashing into the road curb/barrier. Nevertheless, the No Defense setting is even worse than LD^3-NaiveAR, where the vehicle is simply deviated to fall off the road in SLR-Low and SLR-High. Snapshots of the vehicle stopping locations in SF-Straight are shown in Fig. 5.9. The demos of the 4 simulation scenarios and 3 defense settings are available on our project website.

Table 5.3: Maximum deviations to lane center and attack consequences under different defense settings in the 4 simulation scenarios in §5.6.1. Each setting was run for 10 times with randomized attack starting times. Benign driving with LD^3 is also presented and was run for 10 times. The maximum deviations are represented as (mean, std) in meters.

Simulation scenario	Lane straddle dev	Attacked						Benign	
		LD^3		LD^3-NaiveAR		No Defense		LD^3	
		Max dev	Consequence	Max dev	Consequence	Max dev	Consequence	Max dev	Consequence
SLR-Low	0.83	0.47, 0.08	Stop in lane	1.69, 0.06	Stop w/ lane straddle	7.94, 0.05	Fall off road	0.07, 5e-5	Reach destination
SLR-High	0.83	0.69, 0.06	Stop in lane	1.64, 0.16	Stop w/ lane straddle	7.93, 0.04	Fall off road	0.07, 5e-5	Reach destination
SF-Straight	1.00	0.67, 0.23	Stop in lane	1.02, 0.01	Hit curb	1.84, 0.16	Hit tree or barrier	0.14, 7e-4	Reach destination
SF-Curvy	0.75	0.43, 0.14	Stop in lane	0.90, 0.12	Hit lane divider	0.97, 0.14	Hit lane divider	0.31, 0.01	Reach destination

Figure 5.9: Simulation snapshots of the vehicle stopping locations under the 3 defense settings in SF-Straight.

Figure 5.10: Side-by-side views of the AD development chassis and a Toyota Camry.

5.6.2 Evaluation on AD Development Chassis

Experimental setup. We experiment on an AD chassis as shown in Fig. 5.10, which is specifically designed for Level-4 AD system prototyping and testing. The chassis is of a real vehicle size, capable of closed-loop control, and fully equipped with Level-4 AD sensors including LiDAR, GPS, IMU, cameras, RADARs, and ultrasonic sensors. Since AD vehicle testing is not allowed to be on public roads by default, we reserve a parking lot in our institute for the experiments. Specifically, we mark a straight traffic lane with 3.5 m width (the most common lane width in KAIST dataset and our night-time driving trace) in the parking lot and create the corresponding semantic map for Autoware.

Table 5.4: The detection, maximum, and stopping deviations in the three settings at two different driving speeds. We repeat the experiments of *w/* LD^3 *w/ attack* 3 times and report (mean, std) deviations. We do not repeat the other two settings as they are quite stable.

Speed	w/ attack				w/o attack	
	w/ LD^3			w/o LD^3	w/ LD^3	
	Det dev	Max dev	Stop dev	Max/Stop dev	Max dev	Stop dev
4 m/s	0.07m, 0.01m	0.36m, 3e-3m	0.05m, 0.05m	2.59m	0.13m	8e-3m
2 m/s	0.02m, 2e-3m	0.27m, 0.04m	0.01m, 1e-3m	2.23m	0.11m	7e-3m

We ported LD^3 to the Autoware AD system [213], which is currently supported by the AD chassis. To facilitate the attack, we apply the same *FusionRipper* attack trace used in §5.5.4 and §5.6.1 to the localization outputs in Autoware. Unlike OpenPilot and Baidu Apollo, the lane detector in Autoware can only detect lane lines in pixels rather than in the world coordinates. Therefore, we directly obtain the ground truth lane line information from the map using the unmodified localization outputs, since LD is already a mature technology (§5.3) and has been shown to be quite accurate in §5.5 and §5.6.1. We enable the relevant components in Autoware including localization, global/local plannings, and control. During the experiments, the AD chassis is completely driven by Autoware unless taken over by us from a remote controller in emergency situations. We evaluate three defense settings: (1) *w/ LD^3 w/ attack*, (2) *w/o LD^3 w/ attack*, and (3) *w/ LD^3 w/o attack*. For each, we experiment in driving speeds of 2 m/s (4.5 mph) and 4 m/s (9 mph) for safety concerns. We prolong the AR stage by using deceleration $<3\ m/s^2$ in both cases to better showcase the driving behaviors during AR. Specifically, we repeat the experiments for 3 times for *w/ LD^3 w/ attack*. Since the other two are always quite stable, we thus do not record more iterations for those experiments.

Results and demos. Table 5.4 shows the detection, maximum, and stopping deviations under the three settings. As shown, LD^3 on average can detect the attack when the vehicle's physical deviation is still small and start the AR stage. Within the AR period, the average maximum deviations are 0.36 m and 0.27 m at speeds of 4 m/s and 2 m/s, respectively,

Table 5.5: Maximum physical deviations can be achieved without being detected under various LD fluctuation assumptions. Percentages indicate the probabilities of such fluctuations.

Trace	LD fluctuation (μ, σ)	Max physical world deviation		
		0 (100%)	μ (50%)	$\mu + 3\sigma$ (0.3%)
ka-local31	0.12m, 0.08m	0.7m	0.82m	1.06m
ka-local33	0.14m, 0.10m	0.7m	0.84m	1.14m
ka-highway36	0.29m, 0.10m	0.7m	0.99m	1.29m
ka-highway18	0.20m, 0.11m	0.7m	0.90m	1.23m

and the final stopping deviations are always within 0.1 m. In comparison, without LD^3, the vehicle keeps deviating, and we have to manually press the emergency button on the remote to prevent it from crashing into the curb. Such distinctive driving behaviors with and without LD^3 are consistent with our trace-based (§5.5) and simulation results (§5.6.1). Without the attack, the vehicle's trajectories well align with the road centerline (i.e., the reference trajectory Autoware plans to enforce) and eventually complete the route and stop at the center of the lane. We also record demo videos of the vehicle driving behaviors under the three settings (videos are available on our website). As an illustration, Fig. 5.1 visualizes the driving trajectories in the bird's eye view and shows the snapshots of final stopping positions at driving speed of 4 m/s.

5.7 Evaluation against Adaptive Attacks

In this section, we take a step further to examine LD^3's capability under potential adaptive attacks, including (1) an idealized stealthy attack that can evade the detection, and (2) the latest LD-side attack, which is the inherent new attack surface introduced by LD^3 approach (§5.4.1).

5.7.1 Stealthy Attack Evaluation

In this evaluation, we analyze the maximum lateral deviations that a hypothetical stealthy attack can achieve by assuming stronger and unrealistic attack capabilities.

Evaluation methodology. Based on the CUSUM anomaly detection formulation (§5.4.3), the attack should satisfy $S_{i-1} + |D_i^{\text{MSF}} - D_i^{\text{LD}}| - b < \tau$ in order to prevent detection. Assuming the last CUSUM statistic $S_{i-1} = 0$, the maximum MSF lateral deviation without being detected is thus $D_{i,max}^{\text{MSF}} = D_i^{\text{LD}} + \tau + b$, which is also the maximum physical world deviation given the control assumption (§5.2). Since τ and b are fixed in the defense, the attacker can carefully select a timing where the LD has a large lateral deviation fluctuation to the actual vehicle location due to detection noises, and apply the MSF lateral deviation to the same direction as the LD's fluctuation direction to achieve a large physical world deviation. Therefore, the attacker's capability on capturing a particular LD fluctuation window determines the maximum physical world deviations she can achieve without being detected. Thus, we evaluate the maximum physical world deviations by assuming various levels of LD fluctuations that the attacker can capture.

Assumptions on attack capabilities. In this evaluation, we assume the attacker has very unrealistic attack capabilities in order to achieve such a stealthy attack. In particular, the attacker should have a *white-box knowledge* on (1) where exactly on the road that the LD will have a large fluctuation and how much it is, and (2) the attack detection method and parameters used in the target AD system. Moreover, the attacker should also have *precise* and *instantaneous* control over the lateral deviations in the MSF localization outputs in order to execute such attack when large fluctuations appear.

Results. Table 5.5 shows the maximum physical world deviations that the stealthy attack can achieve under different LD fluctuation assumptions. Specifically, we calculate LD fluctuation distributions in each trace and assume that the attacker knows where a certain

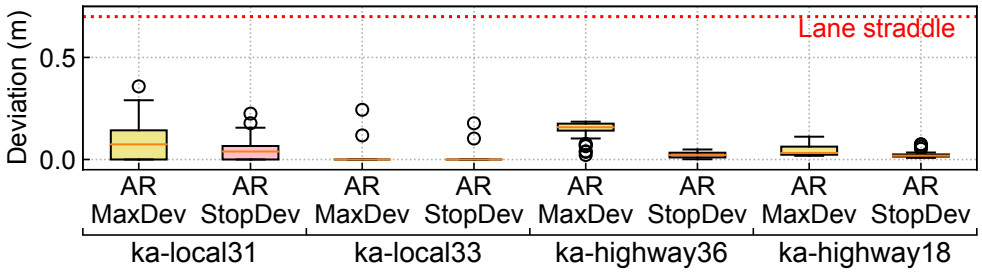

Figure 5.11: Maximum and stopping deviations during the AR period under the LD attack.

level of fluctuation happens. Without any such assumptions, the attacker can at most inject $\tau + b = 0.7$ m lateral deviation, which is just about to touch the lane boundaries. On the other hand, the attacker can *at most* cause 0.99 m and 1.29 m lateral deviations on the 4 traces if she can capture an average and a 3-σ LD fluctuation, respectively. Note that the probabilities of such fluctuations to appear are 50% and 0.3% according to the normal distribution. In conclusion, even under very unrealistic attack assumptions, the maximum lateral deviations are still less than the local road attack goal (1.3 m) for *FusionRipper*, which shows that LD^3 is quite effective at bounding the lateral deviations. Moreover, it also highlights that LD is indeed a mature technology (§5.3) suitable for defense given its high stability.

5.7.2 LD-side Adaptive Attack Evaluation

Evaluation methodology. We explore the defense capability of LD^3 against the latest LD attack in production low-level AD systems, named Dirty Road Patch (DRP) attack [327], which is designed to affect the detected lane line shapes to mislead the automated lane centering system to drive the vehicle out of the lane boundaries. In LD, the lane line shapes are represented as polynomial functions, which are used in LD^3 to calculate the vehicle's lateral deviations (Appendix E). From the 40 attack traces used in the DRP attack paper, we extract the attacked lane line polynomials in each frame and calculate an averaged LD

deviation trace. In LD^3 design, LD attacks cannot disrupt the driving behaviors before the attack is detected since only MSF outputs are used for navigation at this moment. To cause vehicle deviations, the LD attack has to trigger the detection in the first place and affect the *fused* localization in the AR period (§5.4.4) in order to affect the vehicle control. Therefore, we focus on the AR period in our evaluation. To model the DRP attack effect, we apply the deviation trace (start from the detection deviation 0.7 m) to the LD side in the KAIST traces. Since the MSF side is benign and should generally well-align with the physical positions of the vehicle, we set the MSF outputs in the AR period with the same deviation as the fused localization, but to the opposite direction based on the control assumption (§5.2).

Results. Fig. 5.11 shows the maximum and stopping deviations in KAIST traces. As shown, none of them is able to even cause lane straddling. On average, the maximum and stopping deviations in the AR period are only 0.08 m ($\delta = 0.08$ m) and 0.02 m ($\delta = 0.03$ m), respectively. Such a result indicate that LD^3 is quite robust to adaptive attack to the LD side as well. This is because the safety-driven fusion (§5.4.4) in LD^3 can effectively penalize the more aggressive source in the driving context, which in this case is the attacked LD outputs, and prevent the fused localization from being influenced by it.

5.8 Limitations Discussion

Defense coverage of lane detection. In this work, we are the first to explore the novel usage of LD for defense. However, as a defense relying on LD, a potential limitation is the lane line marking coverage. However, as we analyzed in §5.4.2, the non-deterministic nature of attacks to MSF localization greatly alleviate such a limitation, where an LD-based defense has the potential to defend against the majority (99.2%) of the attack attempts. In addition, for important AD applications such as autonomous trucks, they are naturally not subject to such limitation as they mostly operate on highways [38, 46]. At design level, since

high-level AD systems come with semantic maps with accurate road geometry information, LD^3 knows exactly where are the regions without lane line markings and can temporarily disable the defense in such regions (§5.4.3). To address this limitation, a potential future improvement is to also consider other road markings available in such regions, e.g., stop lines [246] and crosswalk markings [94] in intersections, to help localize the vehicle and to detect MSF deviations. Nevertheless, it is unclear how prevalent such road markings are and how mature and robust the existing perception algorithms are to recognize such road markings.

Simultaneous attacks to MSF and LD. Since LD^3 leverages LD to detection lateral-direction attack on MSF, attacks that simultaneously target MSF and LD can thus potentially bypass our detection. In fact, such a vulnerability is a general limitation for CPS security research that uses sensor cross-checking/fusion for defense purposes [162, 161, 74, 356, 217, 233]. However, in practice, the defense value of LD^3 highly depends on *whether such a simultaneous attack already exists or can be easily achieved*. For MSF and LD, neither of them holds today, since (1) although individual attacks on MSF or LD exist, no existing work shows that they can be effectively *coordinated and synchronized* to achieve simultaneous attack effect control, and (2) it is far from trivial to achieve this with existing individual attack vectors. Specifically, among the attack vectors on camera [302, 409, 283, 328, 224, 327, 212, 210], only three works [283, 327, 210] actually evaluated and shown attack effectiveness on LD in realistic AD settings. All these three works consider adding malicious patterns to the ground (e.g., via road patch or stickers) as the attack vector. However, considering the non-deterministic nature of the existing high-level localization attacks (§5.3), it would be hard, if not impossible, for the attacker to figure out where to place the attack pattern beforehand, not to mention how to carefully synchronize the malicious pattern with the localization-side attack to effectively bypass LD^3. Therefore, we consider such simultaneous attack design neither already exists nor can be easily achieved, and leave the systematic exploration of its feasibility as a future research direction.

5.9 Summary

In this work, we perform the first systematic exploration of the novel usage of lane detection (LD) to defend against lateral-direction attacks in high-level AD localization. We design the first domain-specific LD-based defense approach, LD^3, that is capable of both real-time attack *detection* and *response*. Our evaluation on real-world AD sensor traces show that LD^3 is much more effective than directly-adapted physical-invariant based defenses at attack detection with accurate and timely detection. We also show that LD^3 can safely stop the vehicle in the current lane upon detection. We implement LD^3 on two open-source high-level AD systems and evaluate its effectiveness under end-to-end driving with closed-loop control in both simulation and the physical world. We also evaluate against two adaptive attacks and find that LD^3 is robust to an idealized stealthy attack that aims to evade detection and the latest LD-side attack targeting the response stage.

Chapter 6

Defense Opportunity against CV Data Spoofing Attacks

6.1 Introduction

The Connected Vehicle (CV) technologies enable Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications and can help the vehicles and infrastructure make more informed driving/control decisions. CV technologies are attractive as they can benefit our transportation system by improving mobility, reducing safety risks and greenhouse gas emissions, etc. For example, prior studies show that CV technologies can cut vehicle congestion delays by more than a third [52] and help reduce up to 80% of crashes on the road [51]. To enjoy such benefits, government agencies across the globe are competing to push for CV deployments [381, 325, 153]. Particularly, the US is one of a few early adopters that has been testing the CV applications in US cities since 2016 [381, 380, 382, 291]. In general, CV applications can be broadly categorized into *vehicle-side* and *infrastructure-side* applications. The vehicle-side applications aim to maximize fuel efficiency, enable better perception, etc.

A popular example is vehicle platooning [98]. The infrastructure-side applications focus on improving the scheduling or control decisions that will be executed in the infrastructure. The most representative infrastructure-side application is CV-based traffic signal control, which is designed to improve traffic mobility by assigning signal timing plans that prioritize traffic lanes with longer queues in order to reduce the total vehicle delays. Due to the enormous efforts required for CV deployment, the design of CV applications also needs to consider an inevitable transition period where CVs and Regular Vehicles (RVs) coexist on the road. Such a transition period is projected to take over 20 years to reach 95% market penetration rate (i.e., over 95% of total vehicles are CVs) [385].

The widespread deployment of CV applications has a significant impact on the traffic safety and operations, thus making them to be valuable targets of cyberattacks. In this work, we focus on the attacks to *infrastructure-side CV applications* due to their scale of impact. Among them, the most representative one is the recent work on *real-world* infrastructure-side CV application [119], where they discover that real-world CV-based traffic signal control is vulnerable to data spoofing attacks. The authors assume that the attacker can compromise the On-Board Unit (OBU) on a CV to send malicious vehicle states to the signal controller to cause traffic congestions in the intersection. To exploit the vulnerabilities, they design two congestion attacks that target the full deployment and transition periods respectively. The two attacks are demonstrated to be very effective on the USDOT Intelligent Traffic Signal System (I-SIG) [185], a system already under testing in real-world intersections in US cities.

To defeat such data spoofing attacks on infrastructure-side CV applications, in this work, we explore a general spoofing detection strategy that cross validates the cyber-layer vehicles states using the physical-layer ones to identify the spoofers. In our design, we use the readily available *infrastructure-side sensors* [137, 190, 367, 358, 138, 309, 211, 404] to obtain the physical states of CVs. However, the infrastructure-side sensors suffer from a fundamental limitation in the detection range compared to the CV communication range. This leads

to challenges in the defense when dealing with CVs and RVs out of the sensor range. To address them, we leverage a well-established technique in transportation systems, called *traffic models*, which are empirically tuned models from real-world driving data to describe the vehicle driving behaviors in specific traffic conditions. We apply the traffic models as the *traffic invariants* to estimate the physical states of the vehicles out of sensor range. In addition, to find the traffic context of the RVs out of sensor range, we rely on the sensor frames in a future time window to identify the surrounding vehicles of the RVs.

We propose a two-step detector with a Trust Assignment and a Remove-and-Rerun step leveraging the infrastructure-side sensors and traffic invariants. In the Trust Assignment step, we calculate a suspicious score for each CV based on the distance between the reported states and the physical states. Next, we rank the CVs by their suspicious scores, and exclude the most suspicious ones individually and re-execute the I-SIG to confirm their impact to the system. We then apply a threshold-based anomaly detection design to identify the CVs that have large negative impacts to the signal plan.

We evaluate the detector in an industrial-grade traffic simulator, PTV Vissim [311], the same as in the congestion attacks [119]. To ensure the validity, we reproduce the two congestion attacks and find that they have similar attack performance as reported in [119]. In our evaluation, we start by assuming an offline detection setup. First, we examine the effectiveness of the Trust Assignment step and find that it is quite accurate at assigning the suspicious scores, where it always ranks the attack CVs among the Top-5 most suspicious ones. Next, we evaluate the performance of the complete detection pipeline. Our results show that our detector can strike a good balance between the True Positive Rate (TPR) and False Positive Rate (FPR)–it can achieve at least 95% TPRs under all penetration rates while maintaining a low FPR of 7%. Specifically when CVs are fully deployed, our detector shows a perfect detection with 100% TPR and 0% FPR. We also compare our performance with USDOT's official Misbehavior Detection Tool designed for CV applications. Moreover, we evaluate

the robustness of our detector to the infrastructure-side sensor detection noises. The results indicate that our detector can tolerate even 3× normal sensor detection noises.

In our evaluation, we also systematically explore the online detection capability of our detector. We first calculate the required detection timeliness for online detection. To do that, we measure the detector's timing overhead on an embedded device and then limit the future time window to ensure that the whole detection process can finish within the timeliness requirement. Results show that our detector is still effective in the online detection setting, where in the worst case, the FPR is only increased by 5% when the TPR maintains at 98.1% compared to offline detection.

In summary, this work makes the following contributions:

- We explore a general spoofing defense for infrastructure-side CV applications based on the discrepancy between the cyber-layer and physical-layer vehicle states leveraging infrastructure-side sensors. We address the fundamental range limitation in infrastructure-side sensors by applying traffic invariants to infer the CV and RV states.
- We implement the detector for CV-based intelligent traffic signal systems and evaluate the detection effectiveness against two congestion attacks. Our evaluation shows that the detector is quite effective, with at least 95% true positive rate when the false positive rate is below 7%. We also find that our detector is robust to sensor noises.
- We measure the detection overhead on a small embedded device and systematically explore the detection timeliness and effectiveness when it is deployed as an online detector. Our results show that in the worst case, this detection only increases the false positive rate by 5% while maintaining a high true positive rate.

6.2 Threat Model

In this work, we assume a similar threat model as the congestion attacks proposed by Chen et al. [119], where the attacker is able to compromise the in-vehicle CV communication device, i.e., the On-Board Unit (OBU), in their own vehicle to send malicious BSM messages. We do not assume that the attacker can spoof the vehicle identifier in the BSM messages, which are protected and enforced by the Security Credential Management System (SCMS) [379]. Therefore, the attacker needs to use the original certificate associated with the physical vehicle in order to get the messages correctly authenticated.

6.3 Defense Challenges

Data spoofing attacks in CVs are essentially cyber-layer attacks, where attackers send dishonest information of their physical states over the communication channels. Thus, a natural strategy to detect such attacks is to use physical-layer information to validate the cyber-layer information. One readily available physical-layer information source is the infrastructure-side sensors, e.g., cameras [137, 190, 367, 358] and LiDARs [138, 309, 211, 404], which perceive the vehicle positions and speeds in the physical world and thus can serve as the ***physical root-of-trust*** for our detection. Currently, the infrastructure-side sensors are already performing vehicle detection and tracking tasks for red-light enforcement [195] and traffic monitoring [365]. Thus, in this work, we aim to reuse them as a cost-effective solution for data spoofing detection.

Fundamental limitation of infrastructure-side sensors: detection range. Although infrastructure-side sensors can provide accurate detection of the CVs to validate their reported states, their detection ranges are often much more constrained than the CV communication ranges. For example, the effective detection ranges of traffic cameras are usually

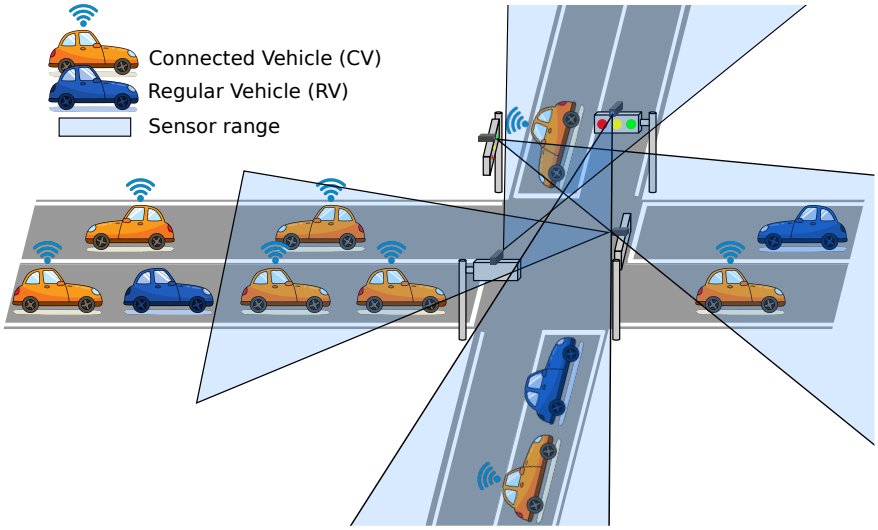

Figure 6.1: Infrastructure-side sensor range limitation. The CV communication range is often much larger than the infrastructure-side sensor range.

at ∼100 meters [190, 358, 254, 132], while CV communication channels (e.g., DSRC and C-V2X) can cover much larger ranges (typically >300 meters [148]). This thus leaves opportunities for the attackers since they can simply spoof CV locations beyond the sensor detection range to evade direct detection. For example, the spoofed CVs are usually located at the end of each intersection approaches with a distance of ∼300 meters to the center of the intersection in the congestion attacks [119]. In fact, this is more of a fundamental limitation of sensors compared to cyber-layer communication–extending the sense range (e.g., installing and synchronizing with additional sensors) is often much more costly and difficult than extending cyber-layer communication ranges (e.g., using signal relay devices or opting to longer range communication protocols such as C-V2X). Because of this fundamental limitation, two defense challenges need to be addressed in order to leverage the infrastructure-side sensors for effective data spoofing detection in the CV context.

Challenge 1: How to systematically propagate the trust from the sensor range to the CV communication range? With the help of infrastructure-side sensors, it is straightforward to verify the reported states of the CVs within the sensor range and establish trust for the ones that match the detection results. But for the CVs outside the sensor range,

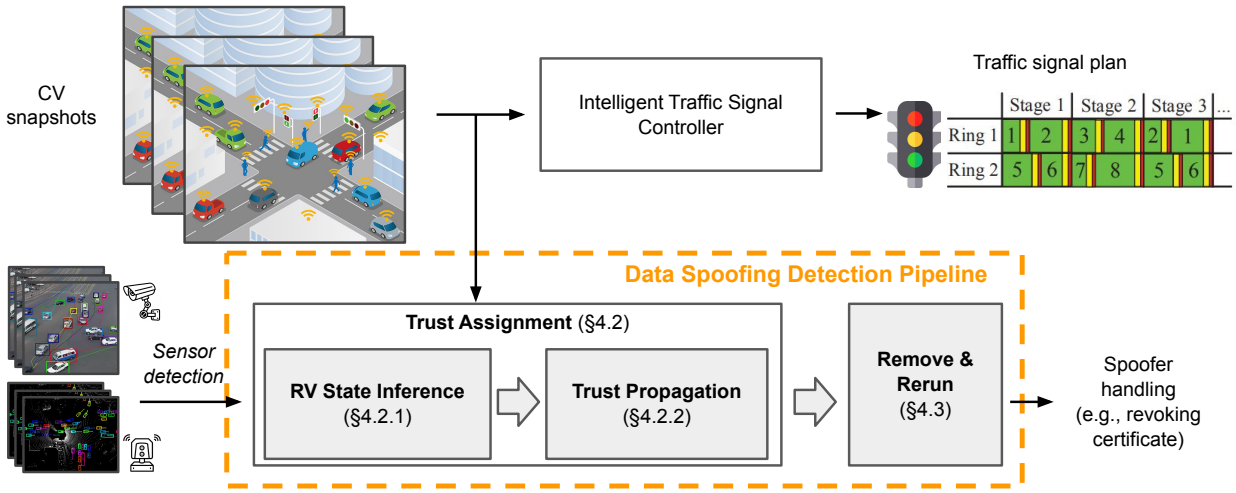

Figure 6.2: Defense design overview.

there is no direct way to measure their states. Despite that, the trusted CVs within the sensor range can provide useful information to verify the positions of the farther CVs, and hence a systematic solution is required to facilitate such trust propagation.

Challenge 2: How to infer the RV states outside the sensor range? Since it is estimated that the deployment of CV technology needs more than 20 years to reach at least 95% market penetration rate [385], there will be an inevitable transition period where CVs and RVs co-exist on the road. During such period, the RVs that are outside of sensor range may disrupt the trust propagation and cause mis- or false detections since the detector is not aware of these RVs. Thus, gaining the knowledge of these RV states are important for accurate spoofing detection.

6.4 Defense Design

In this section, we detail the defense design that addresses the aforementioned challenges.

6.4.1 Design Overview

In our design, we define the *trust* of a CV based on its integrity, i.e., CVs that report a state far away from its ground truth state will be assigned with lower trust (or higher suspicion). Our detector measures the trust of each CV in a traffic snapshot (i.e., the received CV states that a CV application used for decision-making) and pinpoint the ones that have the lowest trust and the largest impact on the CV application performance. As shown in Fig. 6.2, the detector takes the CV snapshot and the corresponding sensor detection results as input, and outputs the suspicious CVs that are likely to be spoofers for further handling. The detection process involves two major steps: *Trust Assignment* and *Remove-and-Rerun*.

Trust Assignment (TA). In this step, we start from our physical root-of-trust, i.e., sensor detection results, to assign *suspicious scores* to the CVs in the sensor range by comparing their reported states with the detection results. Next, we propagate the trust out to the CV range in the order of CVs' reported distances to the sensor range. Since there is no direct way to measure the physical states of CVs out of the sensor range (Challenge 1), we estimate the CV states based on our ***traffic invariants***, i.e., the traffic models, which are empirically derived mathematical equations describing the vehicle driving behaviors under various traffic conditions (§2.1.3). For example, the car-following models can be used to estimate a vehicle's spacing and velocity based on its leading vehicle. We then use the estimated state as a proxy to the CV's ground truth state to calculate the suspicious score.

Since traffic model's accuracy depends on the availability of surrounding vehicle information, it is thus imperative to address Challenge 2 to infer their states in the current CV snapshot. To achieve that, we look into the "future" sensor frames when an RV first enter the sensor range to learn which vehicles are its neighbors. Based on the neighboring vehicles in a future time window, we can thus apply traffic models to infer an RV's state in the current CV snapshot. When the detection process is deployed as an offline analysis, such "future"

sensor frames are always available. However, when deployed as an online detection, we have to delay the detection for certain duration to wait for more sensor frames to come. In this case, since the future time window is limited, the number of recoverable RVs will also be reduced. However, in practice, even a relatively short future time window (e.g., 6 seconds as will be shown in §6.7.2) is sufficient to cover the majority of the RVs. Although the detection is delayed in the online analysis, this is not much a problem for traffic signal control since the attack effect often takes time to build up, e.g., the I-SIG congestion attacks take ∼18 minutes to build up the congestion effects [119].

After the trust assignment, we then rank the CVs based on their suspicious scores. As will be shown in §6.5.2, the spoofing CVs are always ranked with top suspicious scores. In practice, we can aggregate the suspicious score rankings from multiple CV snapshots to more accurately pinpoint the spoofing CVs. Nevertheless, in our design, we apply a Remove-and-Rerun step to further improve the detection accuracy.

Remove-and-Rerun (RnR). In RnR, we re-execute the CV application *with* and *without* a suspicious CV to confirm its impact on the attack objective. The intuition behind this is that the attacker's goal is to disrupt the CV application to cause adverse effects on some CV application metrics, which can often be quantified in the application itself. For example, the congestion attacks on I-SIG are designed to increase the total delay of the vehicles in the intersection, which is exactly what I-SIG is optimized for. Such attack objective driven approach can effectively distinguish attack CVs from the benign ones among the most suspicious CVs.

6.4.2 Trust Assignment

The TA step essentially checks the cyber-layer CV states against their physical-layer states to locate dishonest CVs. It calculates a suspicious score s_i for each CV in the current snapshot

as follows:

$$\begin{aligned} s_i =& \|x_i^c - x_i^p\| \\ x_i^c \in& X^c, x_i^p \in X^p, i \in \{1, ..., n\}, \end{aligned} \tag{6.1}$$

where x_i^c and x_i^p represent the reported and physical states of a CV (detailed later); n is the total number of CVs in the current snapshot; $\|\cdot\|$ denotes the absolute difference between two states, e.g., Euclidean distance. Based on the suspicious scores, we then rank the CVs to obtain the top-K suspicious ones X':

$$X' = \underset{X' \subset X^c, |X'|=K}{\arg\max} \{s_i \mid i = 1, ..., n\}. \tag{6.2}$$

In our design, we define the vehicle state as a 5-tuple vector $x = [t, l, r, v, h]$, where t is the timestamp; l is the traffic lane ID (Fig. 2.6 shows an example of intersection annotated with lane IDs); r is the distance to the intersection center; v is the vehicle speed; h is the vehicle heading. Since traffic signal controllers need to know the intersection geometry in order to plan for dynamic traffic situations, they often have a pre-built map that supports querying these state elements from vehicle BSMs. For example, I-SIG by default will convert the BSMs into such information before optimizing for the signal plan.

Among the cyber- and physical-layer states, x_i^c is readily available since each CV will continuously broadcast its real-time location and velocity. For x_i^p, we rely on the physical root-of-trust (i.e., sensor detection) and trust invariants (i.e., traffic models) to obtain and infer the physical states of CVs in and out of the sensor range. For the former, we do not include concrete designs in this work since vehicle detection and tracking is a well-studied topic in computer vision [254, 132] and already has many commercial products on the market that can provide real-time detection [137, 367, 365, 358]. For the latter, it involves two sub-steps: *RV state inference* and *trust propagation*, and we will detail them in §6.4.2 and

§6.4.2.

Car-following model as the traffic invariant. Since car-following models describe the inter-vehicle spacing that a vehicle will maintain given the leading vehicle's speed, we use them as the traffic invariant to infer the *ideal* position that a vehicle will be located in the current lane based on its leading vehicle. Specifically, we apply the widely-used Newell's car-following model (Eq. 2.4 in §2.1.3) in our design for its simplicity. Given a leading vehicle state $[t, l, r, v, h]$, we can estimate the following vehicle state at time t as follows:

$$\mathcal{M} : [t, l, r, v, h] \rightarrow [t, l, r + (v \cdot \tau + d), v, h]. \tag{6.3}$$

This conforms to the Newell's model that (1) the follower drives at the same speed as the leader, and (2) the follower's spacing is adjusted based on the speed.

RV State Inference

For RVs that are out of sensor range, we infer their states based on their *future* leading vehicles in the sensor range. More concretely, if an RV appears in the sensor detection in any of the future frames between t_0 and $t_0 + T$, we can thus infer the RV's physical state at time t_0 as follows:

$$x_j^{t_0} = \begin{cases} \mathcal{M}(x_{\text{lead}}^{t_0}), & \text{if } \exists\ t \in \{t_0, ..., t_0 + T\}, \|x_j^t - C\| < R \\ \varnothing, & \text{otherwise} \end{cases} \tag{6.4}$$
$$j \in \{1, ..., m\},$$

where x_j is the RV state at time; x_{lead} is the leading vehicle in the sensor frame; m is the total number of RVs; t_0 is the time of the CV snapshot to check for spoofing activity; $\mathcal{M}(\cdot)$ denotes the state estimation function based on the car-following model; C and R are

the geographic center and radius of the sensor range, respectively. Depending on the time window (or the delay) allowed in the detector, it is possible that an RV will not appear in the sensor range. In such case, our detector will simply not be aware of this RV.

Identifying leading vehicle. To find the leading vehicle for a target vehicle, we iterate over all available vehicle states at time t and looking for the one that (1) has the same lane ID, (2) is in front of the target vehicle, and (3) is the closest to the target vehicle. In cases when the distance to the closest leading vehicle is greater than $v_f \cdot \tau + d$, we consider the target vehicle is in free-flow traffic and thus exclude the leading vehicle since it should have negligible impact on the target vehicle's driving behavior.

Handling RVs without leading vehicles. When there is no leading vehicle or the leading vehicle is too far away, we then estimate the RV state at t_0 based on its kinematics, assuming that the RV maintains the same speed between t_0 and t.

Trust Propagation

With more RV states made available out of sensor range, we can now more accurately estimate the physical state of the CVs and propagate the trust from our physical root-of-trust. Similar to the RV state inference, we apply the traffic invariant, i.e., the Newell's car-following model, to estimate the physical state of the CV i based on its leading vehicle as follows:

$$x_i^p = \begin{cases} \mathcal{M}(x_{\text{lead}}), & \text{if } \exists\ x_{\text{lead}} \\ [t_i^c, l_i^c, d_i^c, v_f, h_i^c], & \text{otherwise.} \end{cases} \tag{6.5}$$

Different from RV state inference, even when there is no leading vehicle, we are still aware of the existence of CV i. Thus, instead of ignoring it, we set its state the same as its reported cyber-layer state *except its speed as the free-flow speed of the lane* since we know for sure

there is no leading vehicle and hence the CV should be driving at the free-flow speed in the normal case.

Suspicious score calculation. After we obtain the physical states of the CVs from the sensor detection or from state inference, we then calculate a suspicious score for each CV, which is defined as the *distance* between the cyber- and physical-layer states (Eq. 6.1). More concretely, the suspicious score calculation depends on the availability of physical state elements as below:

$$s_i = \begin{cases} |d_i^c - d_i^p|, & \text{if } \exists\ d_i^p \\ |(v_i^c - v_i^p) \cdot \tau + d|, & \text{otherwise.} \end{cases} \tag{6.6}$$

When the CV's physical distance to the intersection is available, we calculate the suspicious score directly based on the difference to the one in the cyber-layer state. When only the speed element is available in the inferred physical speed, we plug the speed difference into the Newell's model to obtain a spacing penalty such that the suspicious score is a distance measurement and hence comparable to the above case.

6.4.3 Remove and Rerun

We perform the RnR step on all top-K suspicious CVs (Eq. 6.2). Specifically, for each CV k in X', we exclude it from the current CV snapshot and re-execute the I-SIG application to obtain the new total delay. Since the attacker aims to increase the total delay and ultimately cause congestion in the intersection, removing a spoofing CV from the snapshot would likely to reduce the total delay. Formally, we perform the RnR as follows:

$$x_k^c = \begin{cases} \text{spoofer}, & \text{if } \frac{\mathcal{F}(X^c) - \mathcal{F}(X^c \setminus \{x_k^c\})}{\mathcal{F}(X^c)} \geq \epsilon \\ \text{benign}, & \text{otherwise,} \end{cases} \tag{6.7}$$

where $\mathcal{F}(\cdot)$ is the I-SIG application; x_k^c is one of the top-K CVs; ϵ is an empirically determined anomaly threshold for spoofing detection. Since total delay varies under different traffic demands, we calculate a total delay reduction percentage based on the one with all CVs rather than an absolute value.

6.5 Defense Effectiveness Evaluation

In this section, we explore the full potential of our detector against the two congestion attacks [119] in an offline setting, where the detection is performed as a post-processing step on the saved traffic snapshots and sensor frames. Since the detection is offline, we do not enforce any future time window size limitations in the RV state inference (§6.4.2).

6.5.1 Evaluation Methodology

Real-world intersection configuration. In our evaluation, we use the PTV Vissim [311], which is an industrial-grade traffic simulation software, to generate the traffic snapshots for the I-SIG system and execute the produced signal plans. The Vissim software relies on car-following and lane-changing models to generate CV/RV traffics. However, it is worth noting that the car-following model used in Vissim is a much more sophisticated one, named Wiedemann's model [160], than the Newell's model used in our detector design. Unlike deterministic models such as the Newell's, the Wiedemann's model adds randomnesses in each vehicle's behavior [160] to generate realistic and diverse traffic situations. To faithfully reproduce the simulation setup, we obtained the detailed Vissim configurations they used when evaluating the attack. Specifically, we simulate a real-world intersection as shown in Fig. 2.6 and Table 6.1. In addition, we also use the same traffic demand (i.e., vehicle arrival rate) and turning ratio (i.e., vehicle lane changing probability) in each lane that the authors

Table 6.1: Configurations of the real-world intersection (Fig. 2.6) used in this work and in the congestion attacks [119].

Approach	ID	Speed limit	Approach length
Eastbound	1	35 mph	220 m
Southbound	2	30 mph	300 m
Westbound	3	45 mph	300 m
Northbound	4	40 mph	300 m

Table 6.2: Attack performance reported in the congestion attacks [119] and of our reproduced attacks. *PR* is short for penetration rate. *Avg total delay inc* is the average total delay increment caused by the attacks.

CV deploy status	Full deploy		Transition Period					
	Arrival time attack		Queue length attack					
	PR = 100%		PR = 75%		PR = 50%		PR = 25%	
	[119]	Ours	[119]	Ours	[119]	Ours	[119]	Ours
Avg total delay inc	66.7%	52.9%	181.6%	172.8%	193.3%	176.6%	133.2%	147.9%

collected from real-world intersection [119].

Infrastructure-side sensor detection assumption. We assume a typical traffic camera configuration, where there is a camera for each approach similar to Fig. 6.1. In this section, we assume perfect camera detections, i.e., the vehicle positions and speeds can be accurately detected in the sensor range. We will relax this assumption later in §6.6 to evaluate the robustness against sensor noises. We set the sensor range to 91 meters (300 ft) in the evaluation since this is a common specification for traffic cameras [190, 358] and prior works on traffic camera detection also report similar capabilities [254, 132].

Congestion attacks reproduction. We evaluate our detector against the two congestion attacks on the I-SIG system [119]. Since we do not have the original attack traces, we reproduce the congestion attacks and report the attack performance in our Vissim and I-SIG setup to demonstrate the reproduction fidelity. Similar to [119], we also use three random seeds in the Vissim to compensate the randomness. To demonstrate the defense capability, we focus on the strongest attacks in each penetration rate setting. Table 6.2 shows the attack

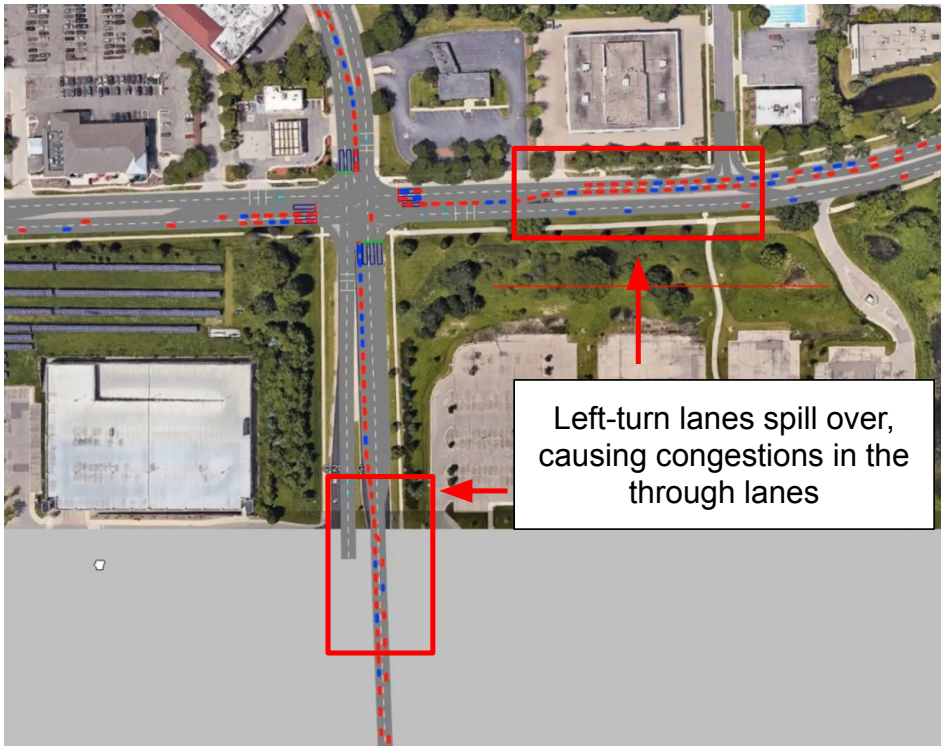

Figure 6.3: A Vissim snapshot of the traffic congestions caused by our reproduced attacks.

performance reported in [119] and the ones reproduced by us. Specifically, the average total delay increment is the extra total delay caused by the spoofing CV compared to the benign scenario. As shown, the attack performance reproduced by us is on par with the ones reported in [119]. We also verify in the Vissim view that the congestion builds up during the simulations. Fig. 6.3 shows a snapshot of the traffic congestions caused by our reproduced attacks.

Baseline detector: USDOT Misbehavior Detection Tool. The USDOT Misbehavior Detection Tool (USDOT MDT) is an official tool developed by the USDOT to detect CV BSM messages that are "inconsistent with the corresponding vehicle's true status, position, or behavior" [378]. Specifically, the USDOT MDT checks for infeasible CV position and speed, incorrect BSM message format and transmission rate, etc. Since the tool has already been passed onto companies such as GM and Ford [378] and may potentially be incorporated in their CV products, we evaluate its spoofing detection effectiveness against the congestion

attacks and use it as a comparison baseline.

Evaluation metrics. To quantify the spoofing detection performance, we show the *True Positive Rates (TPRs)* and *False Positive Rates (FPRs)* under the attacked and benign CV snapshots, respectively. To generate such CV snapshots, we simulate each Vissim seed twice–one under attack and another without any attack. Each simulation lasts for 4000 seconds, which consists of about 70–140 snapshots per seed depending on the congestion level. Particularly in the attacked simulations, the attacker may choose not to spoof a snapshot if she cannot find any spoofing location that can increase the total delay. In addition, we also plot the *Receiver Operating Characteristic (ROC) curves* to systematically show the TPR and FPR variations under different anomaly thresholds ϵ (§6.4.3). Particularly, to validate the effectiveness of trust assignment, we also report the *Top-K rate* to show the percentage of CV snapshots that rank the spoofing CV among the top-K most suspicious CVs.

6.5.2 Results

Detection accuracy of Trust Assignment. Before evaluating the complete detection pipeline, we first look at how effective is the TA at assigning the suspicious scores. Table 6.3 shows the Top-K rates in the attacked snapshots. As shown, the TA step is quite effective at finding the attack CVs, where it *always ranks the attack CV among the top-5 most suspicious CVs across all PR settings*. Therefore, we set $K = 5$ in the following RnR step (thus denoted as RnR-5) as this empirically ensures that the attack CV, if any, is always among the ones that will be validated. Moreover, over 89% of the total CV snapshots rank the attack CVs at Top-1. The reason that some attacked CVs are not ranked at the top is mainly that in these snapshots some benign CVs exhibit driving behaviors that are not considered in the car-following model (e.g., lane changing) such that the TA mistakenly assigns high suspicious scores to these benign CVs. This indicates that a simpler detector that purely rely on TA

Table 6.3: Suspicious score rankings of the attack CVs in Trust Assignment. The numbers in the parentheses are the CV snapshots that we rank the attack CV in Top-K and the total number of CV snapshots in the simulation, respectively.

PR	Attack	Top-1 Rate	Top 3 Rate	Top-5 Rate
100%	Arrival time attack	91.2% (93/102)	99.0% (101/102)	100.0% (102/102)
75%	Queue length attack	89.3% (200/224)	99.1% (222/224)	100.0% (224/224)
50%		90.7% (194/214)	99.5% (213/214)	100.0% (214/214)
25%		92.8% (180/194)	99.5% (193/194)	100.0% (194/194)

is unlikely to be very effective. Yet, the TA is a crucial part in the detection process as it effectively narrows down the detection scope for RnR to accurately pinpoint the attack CVs.

Effectiveness of the complete detector pipeline. We now evaluate the performance of our complete detection pipeline (TA and RnR). Since the benign CV snapshots are also involved in the evaluation, we need to select the anomaly threshold in RnR (§6.4.3) such that it would not incur many false positives while still maintaining a good detection accuracy. Table 6.4 shows the TPRs and FPRs of our detector and the USDOT MDT under different PRs. Specifically for our detector, we list the best TPRs that can be achieved under different FPR levels by varying the anomaly threshold. Systematic analyses on the impact of anomaly threshold will also be shown later. As shown in the table, our detector can achieve a *perfect detection with 100% TPR and 0% FPR* when the CVs are fully deployed (i.e., PR = 100%). Also, when the FPR is 7%, the detector can achieve at least 95% TPRs in all PR settings. Benefitted from the RnR-5, the complete detector pipeline further improves the detection accuracy on top of TA. This is because the RnR-5 directly leverages the attack objective to distinguish attack and benign CVs. As shown in Fig. 6.4, removing an attack CV can reduce the I-SIG total delay much more significantly than removing a benign one in most cases.

Nevertheless, the detection performance in the lower PR settings is generally worse, where the TPR drops to 85.6% when FPR is 5%. The lower performance is mostly caused by the congestion effects in the attacked snapshots, where a benign CV stops in the middle of an empty lane waiting to cut into a queue in the adjacent lane. This makes the queue estimation

Table 6.4: Attack detection performance of our detector (TA & RnR-5) and the USDOT MDT [378]. *PR*: Penetration Rate, *TPR*: true positive rate, *FPR*: false positive rate.

PR	Attack	Ours					USDOT MDT	
		TPR (FPR=7%)	TPR (FPR=5%)	TPR (FPR=3%)	TPR (FPR=1%)	TPR (FPR=0%)	TPR	FPR
100%	Arrival time attack	100%	100%	100%	100%	100%	0%	0%
75%	Queue length attack	100%	99.1%	96.0%	70.1%	61.6%	0%	0%
50%		99.5%	99.1%	98.1%	78.5%	0.5%	0%	0%
25%		95.4%	85.6%	82.0%	69.6%	0%	0%	0%

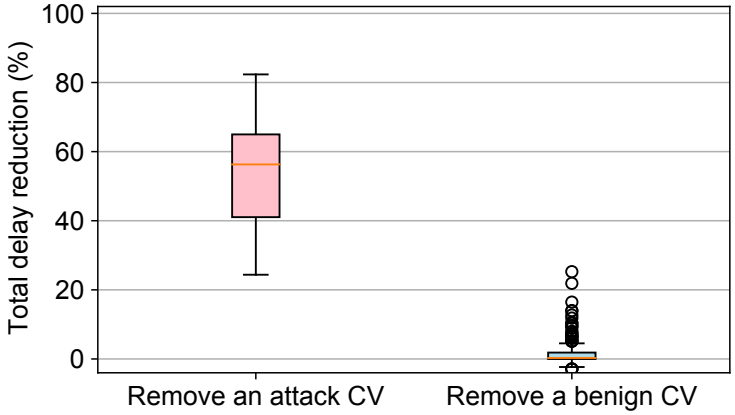

Figure 6.4: Total delay reductions difference between removing an attack and a benign CV.

(not used in full deployment) in I-SIG to mistakenly think that there are many RVs in the empty lane and thus allocates an unnecessary long green time for it. In such cases, removing this benign CV from the snapshot will actually reduce the total delay. In practice, such cases might not be much of a concern since (1) excluding such benign CVs improves the I-SIG performance, and (2) one can use attack detection from multiple snapshots to reduce the false positives. More discussion on this is in §6.8.4.

Effectiveness of the USDOT MDT. As shown in Table 6.4, the USDOT MDT completely fails to detect any attack CVs and thus leads to 0% TPRs. The reason is that the USDOT MDT is designed to identify CVs that report unrealizable trajectories, such as the ones reporting very high speed or very far location in the BSMs. However, in the congestion attacks [119], all spoofed CV locations and speeds are physically feasible, e.g., a CV driving

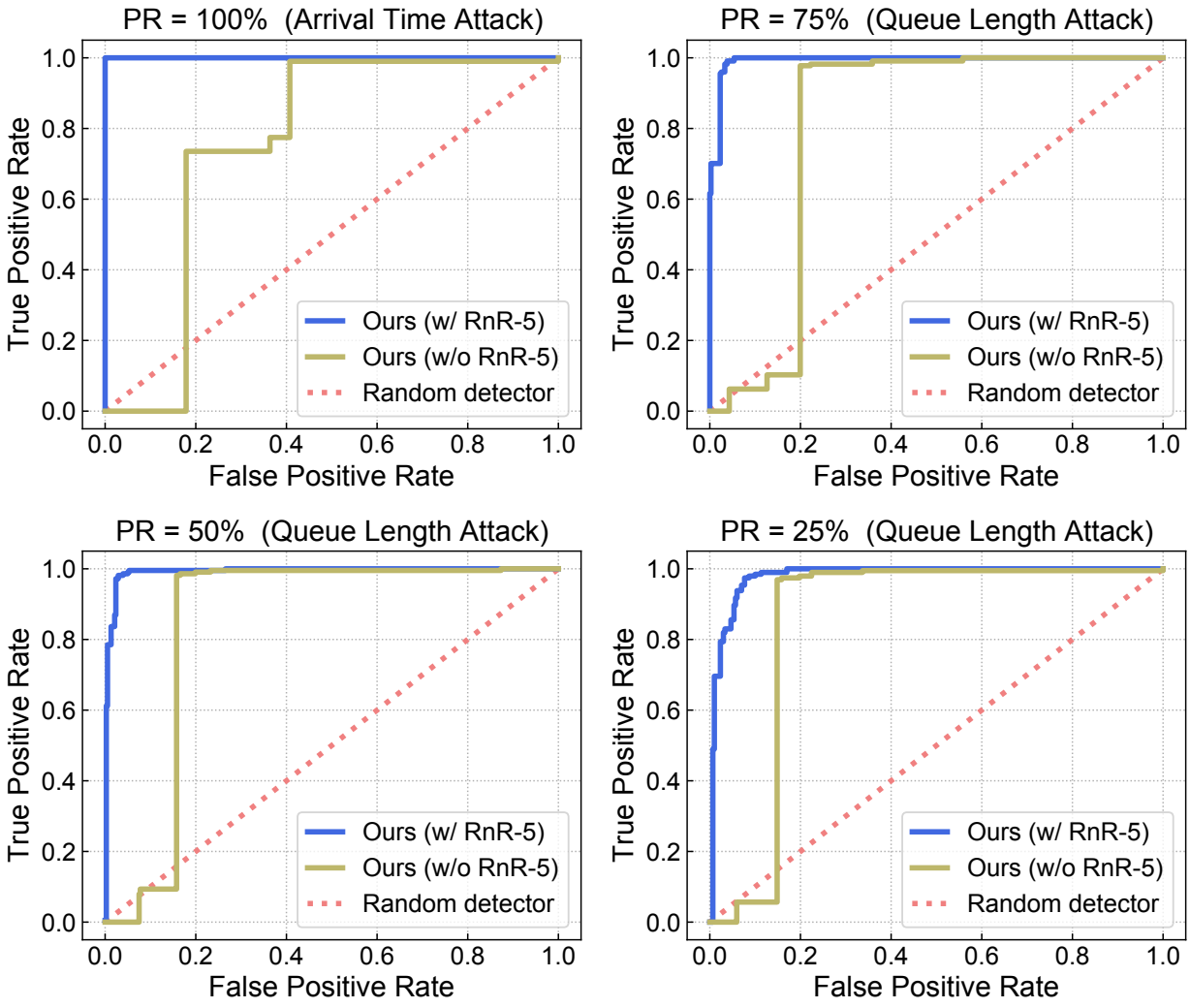

Figure 6.5: Attack detection ROC curves of our detector with and without the RnR-5.

at ∼300 meters away from the intersection with a speed ∼1 m/s.

Detection effectiveness under different anomaly thresholds. To systematically evaluate the detection performance, we plot the detection ROC curves by varying the anomaly thresholds. To highlight the importance of RnR-5, we also plot the ROC curves of a simpler detector design *without* RnR-5. Specifically, we use a threshold-based anomaly detection design to classify CVs as attackers if their suspicious scores are above the threshold. Fig. 6.5 shows the ROC curves of the detector with and without the RnR-5. As shown, incorporating the RnR-5 greatly improves the detection performance in all PR settings. Moreover, the ROC curves indicate that our detector can generally well-distinguish the attack and benign CVs.

6.6 Robustness to Infrastructure-Side Sensor Noises

Although our detector shows high effectiveness in §6.5, we have not considered the impact of practical factors such as sensor noises. In our design, we rely on vehicle detections from the infrastructure-side sensors to build our physical root-of-trust. Thus, any sensor detection noises may affect the later stages in our detector pipeline. Although high-precision sensors such as LiDARs are available in intersections [138, 309, 211, 404], they often cost much more than sensors such as cameras [137, 190, 367, 358, 254, 132]. Thus, in this section, we evaluate the robustness of our detector against camera detection noises to demonstrate the practicality.

Experimental setup. In our TA design (§6.4.2), since we compare the reported CV positions and velocities with the detected positions and velocities in the sensor range to assign suspicious scores, sensor detection noises in the position and velocity will directly affect the suspicious score calculation. A prior work [254] has quantified such errors for traffic cameras,

where they apply computer vision techniques to estimate the position and velocity of vehicles within a certain range to the intersection. Specifically, they discover that the average position errors are only 0.8 meters and 1.7 meters within 50-meter and 120-meter distances from the camera, respectively. And the position errors are mostly longitudinal, i.e., in the direction of the lane. For the velocity error, they find that the estimated speed has an average error of 1.47 m/s to the ground truth vehicle speed. Thus, in our evaluation, we model the camera detection noises based on their findings. More concretely, we inject random errors sampled from Gaussian noise distributions $e_{\text{pos}} \sim N(0, \sigma_{\text{pos}}^2)$ and $e_{\text{vel}} \sim N(0, \sigma_{\text{vel}}^2)$ to the detected position and velocity for all vehicles (CVs and RVs) in the sensor range. We select $\sigma_{\text{pos}} = 1.7$ meters and $\sigma_{\text{vel}} = 1.47$ m/s in the evaluation. In addition, we also evaluate larger noise levels by scaling the error amounts to $2 \times \{\sigma_{\text{pos}}, \sigma_{\text{vel}}\}$ and $3 \times \{\sigma_{\text{pos}}, \sigma_{\text{vel}}\}$, respectively.

Results. Fig. 6.6 shows the detection ROC curves with and without camera detection noises. As shown, the detection performance is barely affected when PR is 100% or 75%, and is only slightly worse in the lower PR settings. This is mainly because the camera detection errors are relatively much smaller than the distance between the spoofed CV location and the location estimated from the traffic invariant. For example, even with 3× position errors, the error standard deviation merely equals to a common vehicle length (4–5 meters). In comparison, to induce large total delay increment, the spoofed CV location is usually >18 meters away from the TI-estimated location.

6.7 Online Detection Exploration

The evaluations above assume an offline detection setup, which is useful to understand the upper bound detection performance. Meanwhile, it is quite attractive if our detector can conduct online detection since we can then effectively prevent the attack CVs from building up the congestion by temporarily excluding these CVs or permanently revoking

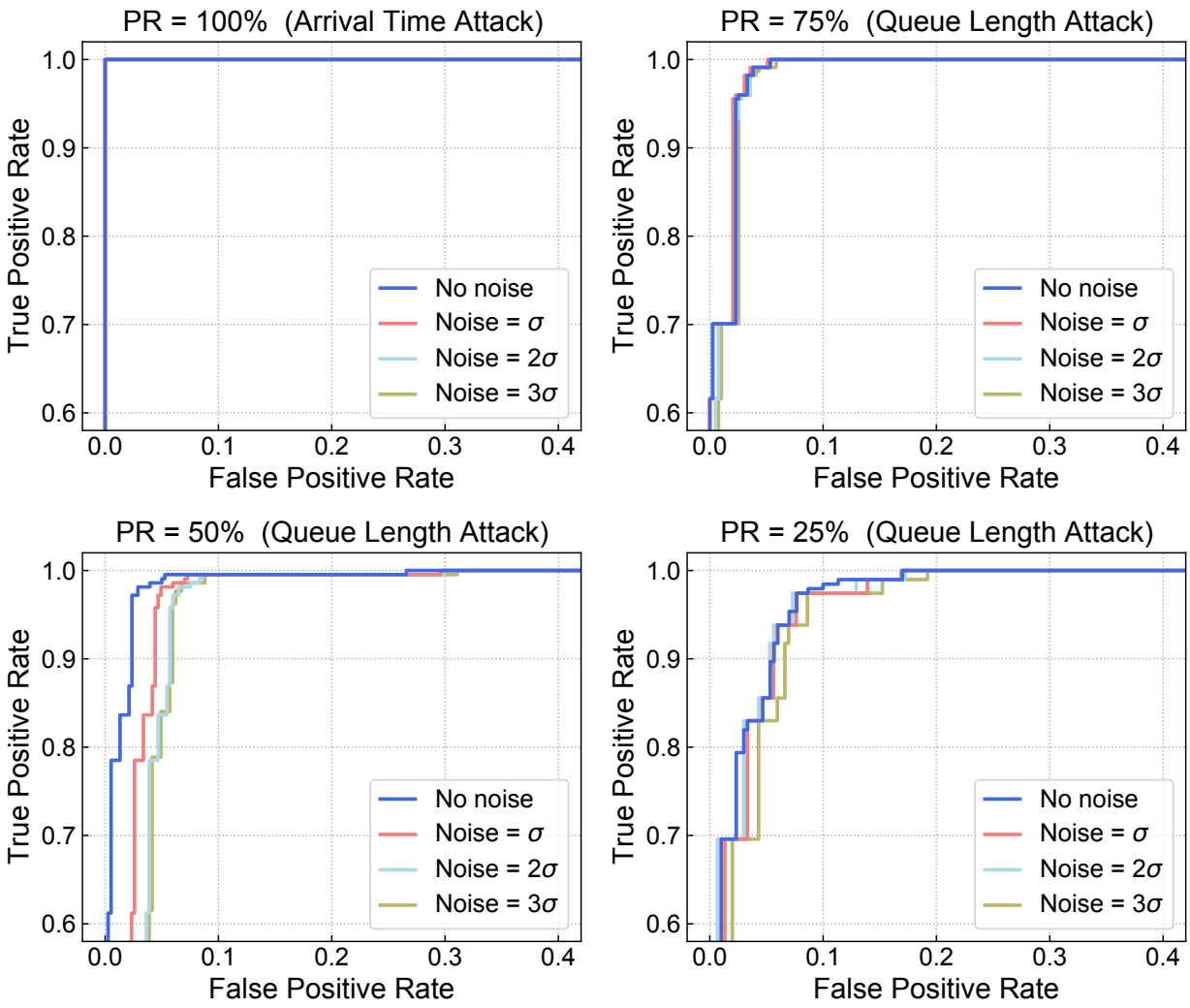

Figure 6.6: Attack detection ROC curves (zoom-in view) under different levels of camera detection noises ($\sigma = \{\sigma_{\text{pos}}, \sigma_{\text{vel}}\}$).

their certificates. Therefore, in this section, we relax the assumption on the future sensor frames and systematically explore the trade-off between detection effectiveness and timeliness in an online detection setup.

6.7.1 Evaluation Methodology

In our evaluation, we first measure the timing overhead of our design and identify the detection timeliness requirements in the traffic signal context. Next, we evaluate the online detection effectiveness under different timeliness requirements by limiting the size of future time window in the detection pipeline.

Timing overhead. We envision that our detector will be running on the Road-Side Unit (RSU) that runs the CV applications. Since the CV deployment requires installing numerous RSUs to provide wide coverage, the device cost would be a large concern for the deployment. Thus, to ensure that we do not assume any unreasonable computation capability, we measure the timing overhead on a Raspberry Pi 3 Model B [317], which is a low-end embedded device that costs only $35. This small computing device comes with an ARM CPU and 1 GB RAM and runs a Debian-based OS. We measure the timing overhead of the two detector steps for 100 times, and their statistics are listed in Table 6.5. As shown, the TA and RnR-5 steps take on average 4.83 sec and 4.48 seconds, respectively. Their timing overhead fluctuates depending on the system load and the number of vehicles in the snapshot. At maximum, they take 9.05 sec and 6.13 sec, respectively. Therefore, we estimate the maximum required duration of the detection process as 15.58 sec. Note that here we did not count the future time window (§6.4.2) that we should wait prior to the detection. For online detection, the overall detection delay should be the sum of the detection timing overhead and the future time window, which we specify based on the timeliness requirements in the next part.

Detection timeliness requirements. The I-SIG system runs at the end of each signal

Table 6.5: Timing overhead of the design components. Results are summarized from 100 measurements.

Components	Mean	Std	Min	Max
TA (§6.4.2)	4.83 s	2.09 s	0.38 s	9.05 s
RnR-5 (§6.4.3)	4.48 s	0.94 s	2.47 s	6.13 s

stage (§2.1.3) to plan for the next signal timing plan. Thus, the online attack detection needs to be finished within one stage in order to keep up with the same pace as I-SIG executions. Specifically, for a common major arterial intersection as we used in our evaluation, the *minimum* green light duration for each phase needs to be configured between 7–15 sec [376]. In our evaluation, we set the minimum green light to 7 sec (which entails the tightest constraint on the timeliness). Since each phase will also go through a yellow light period (typically 3 sec) and a short red clearance period (to accommodate for the potential red light runners, typically 1 sec), the minimum duration for each phase is thus 11 sec. Hence, one signal stage (i.e., two sequential phases) will occupy at least 22 sec. In other words, the overall detection delay should be within 22 sec to achieve online detection. Excluding the detection timing overhead, we thus have 6 sec for the future time window. Therefore, in our evaluation, we set the future time window to 6 sec for online detection.

6.7.2 Online Detection Effectiveness

Fig. 6.7 shows the comparison of the offline and online detection ROC curves. Since the future time window only affects the RV state inference (§6.4.2), online detection has no impact in the full deployment period (PR = 100%) and has little impact when PR is 75% where the number of RVs are limited. Interestingly, limiting the future time window has a larger impact on PR = 50% compared to PR = 25%. This is because when PR = 25%, although the number of RVs increases, the number of CVs decreases correspondingly. Therefore, the detector is less affected due to the smaller number of CVs that need to be validated. Nevertheless, even

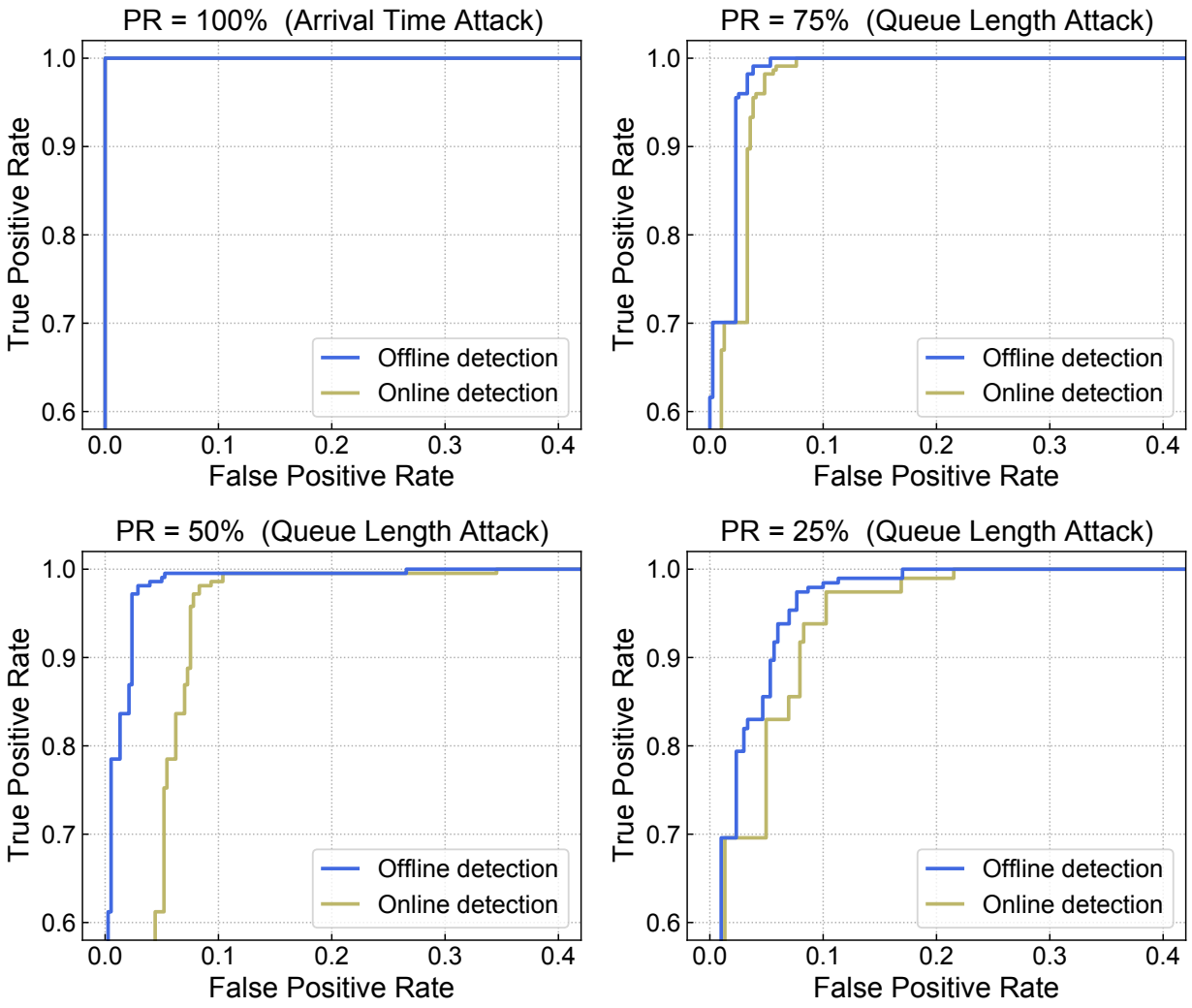

Figure 6.7: Attack detection ROC curves (zoom-in view) of the offline and online detection setups.

when PR = 50%, the FPR only increases by 5% if we aim to keep the same TPR at 98.1%. Such a good online detection performance is likely because a future time window of 6 sec can already cover many RVs to enter the sensor range.

6.8 Discussions

6.8.1 Defense Generality Discussion

In this work, we demonstrate the effectiveness of our detector against two representative congestion attacks on the infrastructure-side CV application. However, since our defense is based on a general principle that uses physical-layer information to cross validate the cyber-layer information, it presents as a promising spoofing defense solution for other types of CV applications and attacks objectives.

Generality to vehicle-side CV applications. There are a wide variety of CV applications proposed in the literature, where many also focus on vehicle-side applications, e.g., cooperative adaptive cruise control [337] and vehicle platooning [98]. For such CV applications, the coverage of infrastructure-side sensors would be a main concern, since the existing infrastructure-side sensors are mostly installed in intersections due to their direct benefit to signal control. However, given that the minimum required signal spacing for major arterial roads is 1/2 miles (800 meters) [384, 272, 342], the intersection sensors can already provide a good coverage to build the physical root-of-trust even for regular urban roads. In addition, increasing efforts are put on installing more roadside sensors for autonomous and connected vehicles recently [70].

Generality to other attack objectives. In our design, the RnR (§6.4.3) is designed as an attack objective driven approach to find the attack CVs. In this work, we focus on the attack goal from existing congestion attacks that target traffic signal control. However, it can also be adapted to other attack objectives, e.g., safety damage, if it is possible to quantify such attack objectives. Intuitively, unless the attacker applies a blind attack without any guidance from the CV application or the vehicle status on the roads, we as the defender can often obtain some indications to facilitate the RnR. Nevertheless, since the TA is quite

effective at narrowing the scope of attack CVs among the most suspicious ones (Table 6.3 in §6.5), we can also aggregate the suspicious score rankings from multiple CV snapshots to accurately pinpoint the attack CVs without involving the RnR.

6.8.2 Alternative Defense Designs

Using CV sensors to extend the infrastructure-side sensor range. One possible way to alleviate the sensor range limitation is to also incorporate the onboard sensors from CVs. This is indeed a promising direction to overcome the limitation. However, in our current design, we do not consider this due to two reasons. First, the CVs may not have any perception sensors that can detect the surrounding vehicles, since the minimum requirement on CVs are the installation of an On-Board Unit (OBU), in which only the GPS sensor is available (e.g., [15]). Second, incorporating sensor detections (self-reported) from CVs opens another door for attackers, which may complicate the detector design.

Using future sensor frames to check the existence of attack CVs that are out of sensor range. In our design, for the CVs that are out of sensor range, we choose to not detect attacks based on their existences in the future time frames (§6.4.2). This is because the attacker is able to spoof the trajectory to a benign looking one without sudden termination after the attack is taking effect (i.e., the spoofed CV location is taken into account in the I-SIG). For example, the attacker can differentiate the CVs and RVs on the road since the CVs will broadcast their real-time positions. Thus, the attacker can overlay the spoofed trajectory onto an RV that is passing by to evade such detection. Moreover, even benign CVs may also trigger such detection, since they may simply divert into a driveway on the roadside without entering the sensor range.

6.8.3 Handling Lane-Changing Vehicles

The Vissim software used in our evaluation is designed to simulation realistic traffics. Aside from car-following simulation, Vissim also simulates the lane-changing behaviors of the vehicles when generating the traffic. Specifically, such lane changing behaviors will mostly affect the car-following relationship of the RVs out of sensor range, since the CVs will continuously broadcast their positions such that the detector is aware of the changes of their car-following relationships. However, in our current design, for RVs that performed lane changing prior to entering the sensor range, we do not attempt to recover their prior lane sequences. Instead, we still use their car-following relationship observed in the sensor range to estimate their states. Such errors may disrupt the suspicious score calculation of the following CVs and hence affect the detection accuracy. One potential improvement is to apply lane-changing models [275] to probabilistically model the RV states in different lanes. However, it is unclear whether including such lane changing models will negatively affect the common case detection, and therefore we leave it as a future direction. Nevertheless, our evaluation results in §6.5.2 show that the current detector design can already achieve a good detection accuracy.

6.8.4 Spoofer Handling

In this work, we do not propose concrete designs to handle the attack CVs since it is an orthogonal problem. In practice, there are generally two ways to handle an attack CV. The first is to temporarily exclude the CV from future CV application executions. This way, we can ensure that the CV, if it is indeed an attacker, will not build up enough total delays to cause congestion effects. However, the attacker will not be punished if such handling is adopted. The second is a certificate revocation based handling. With such an approach, the detector needs to achieve a very low FPR to not mistakenly revoke the certificate of a benign CV. In our design, we show that in certain PR settings, it is difficult to further lower the

FPR without sacrificing the TPR (§6.5.2). However, we can aggregate the detection results from multiple snapshots–if a CV is constantly being flagged as attacker, we then decide to revoke its certificate. In fact, the attacker may choose to use the same CV to attack multiple CV snapshots since the attacker (1) generally needs to spoof many CV snapshots to build up the congestion effects [119], and (2) will likely be using the same CV due to the attack cost (as CV certificates are associated with physical vehicles).

6.9 Summary

In this work, we explore a general defense solution against data spoofing attacks on infrastructure-side CV applications. Building upon the general principle that using physical-layer information to cross validate cyber-layer information, we leverage the readily-available infrastructure-side sensors, such as traffic cameras, to estimate the physical-layer CV states. However, we identify that such infrastructure-side sensors suffer from a fundamental limitation, where the sensor range is generally much smaller than the CV communication range. To address this, we borrow the well-established traffic models from the transportation domain and use them as the traffic invariants to infer the vehicle states that are out of sensor range. We implement and evaluate our detector against two representative data spoofing attacks that aims to cause congestions in the intersections by exploiting the CV-based traffic signal control. Our results show that the detector can effectively identify the spoofer with high detection accuracy and is robust to sensor noises. We also demonstrate that an online detection setting only slightly degrade the detection performance.

Future work. We envision that our defense solution can be generally applied to a wide-range of CV applications and attack objectives, not limited to congestion attacks on infrastructure-side CV applications. Therefore, in the future work, we plan to explore other data spoofing attacks on diverse applications and systematically study the detection effectiveness against

such attacks.

Chapter 7

Systematization of Knowledge in Semantic AD AI Security

7.1 Introduction

Autonomous Driving (AD) vehicles are now a reality in our daily life, where a wide variety of commercial and private AD vehicles are already driving on the road. For example, commercial AD services, such as self-driving taxis [87, 47], buses [11, 200], trucks [43, 204], delivery vehicles [24] are already publicly available, not to mention the millions of Tesla cars [36] that are equipped with Autopilot [33]. To achieve driving autonomy in complex and dynamic driving environments, AD systems are designed with a collection of AI components to handle the core decision-making process such as perception, localization, prediction, and planning, which essentially forms the "brain" of the AD vehicle. However, this makes these AI components highly security-critical as errors in them can cause various road hazards and even fatal consequences [45, 42].

Unfortunately, today's AI algorithms, especially deep learning, are known to be generally

vulnerable to adversarial attacks [352, 175]. However, since these AI algorithms are only components of the entire AD system, it is widely recognized that such generic AI component-level vulnerabilities do not necessarily lead to system-level vulnerabilities [332, 304, 149, 209]. This is mainly due to the large **semantic gaps**: (1) from the system-level attack input spaces (e.g., adding stickers [157], laser shooting [111]) to those at the AI component level (e.g., image pixel changes [352, 175]), or *system-to-AI semantic gap*, which needs to overcome fundamental design challenges to map successful attacks at the AI input space back to the problem space, generally called the *inverse-feature mapping problem* [304]; and (2) from AI component-level attack impacts (e.g., misdetected road objects) to those at the system level (e.g., vehicle collisions), or *AI-to-system semantic gap*, which is also quite non-trivial, e.g., when the misdetected object is at a far distance for automatic emergency braking [149, 332], or the misdetection can be tolerated by subsequent AI modules like object tracking [209]. Thus, for an AI security work to be semantically meaningful at the system level, it must explicitly or implicitly address these 2 general semantic gaps. In this work, we call such research space **semantic AI security** (as opposed to generic AI security), following the semantic adversarial deep learning concept by Seshia et al. [332].

Over the past 5 years, increasingly more research works are performed to tackle the aforementioned semantic AI security challenges in AD context, which started to show an exponential growth trend since 2019 (Fig. 2.9). However, to the best of our knowledge, so far there is no comprehensive systematization of this emerging research space. There are surveys related to AD security, but they either did not focus on AD AI components (e.g., on sensor/hardware [218], in-vehicle network [320]), or touched upon AD AI components but did not focus on the works that addressed the semantic AI security challenges above [142, 312]. Since (1) the latter ones are much more semantically and thus practically meaningful for AD systems, (2) now a substantial amount of them have appeared (over 50 as in Fig. 2.9), and (3) such attacks in AD context have especially high safety-criticality since AD vehicles are heavy, fast-moving, and operate in public spaces, we believe now is a good time to summarize

the current status, trends, as well as scientific gaps, insights, and future research directions.

In this work, we perform the first systematization of knowledge (SoK) of the growing semantic AD AI security research space. In total, we collect and analyze 53 such papers, with a focus on those published in (commonly-recognized) top-tier venues across security, computer vision, machine learning, AI, and robotics areas in the past 5 years since the first one appeared in 2017 (§7.2). Next, we taxonomize them based on research aspects critical for the security field, including the targeted AI component, attack/defense goal, attack vector, attack knowledge, defense deployability, defense robustness, as well as evaluation methodologies (§7.3). For each research aspect, we emphasize the observed domain/problem-specific design choices, and summarize their current status and trends.

Based on the systematization, we summarize 6 most substantial scientific gaps (§7.4) observed based on quantitative comparisons both vertically among existing AD AI security works and horizontally with security works from closely-related domains (e.g., drone [282]). With these, we are able to provide insights and potential future directions not only at the design level (e.g., under-explored attack goals and vectors), but also at the research goal and methodology levels (e.g., the general lack of system-level evaluations), as well as at the community level (e.g., the substantial lacking of open-sourcing specifically for the works in the security community).

Among all these scientific gaps, the one on the general lack of system-level evaluation is especially critical as it may lead to meaningless attack/defense progress at the system level due to the AI-to-system semantic gap (§7.4.1). To effectively fill this gap, it is highly desired to have a community-level effort to collectively build a common system-level evaluation infrastructure, since (1) the engineering efforts for building such infrastructure share common design/implementation patterns; and (2) in AD context, the system-level evaluation results are only comparable (and thus scientifically-meaningful) if the same evaluation scenario and metric calculation are used.

In this work, we thus take the initiative to address this critical scientific methodology-level gap by developing a uniform and extensible system-driven evaluation platform, named *PASS* (Platform for Autonomous driving Safety and Security), for the semantic AD AI security research community (§7.5). We choose a simulation-centric hybrid design leveraging both simulation and real vehicles to balance the trade-offs among fidelity, affordability, safety, flexibility, efficiency, and reproducibility. The platform will be fully open-sourced (will be available on our project website [25]) so that researchers can collectively develop new interfaces to fit future needs, and also contribute attack/defense implementations to form a semantic AD AI security benchmark, which can improve comparability, reproducibility, and also encourage open-sourcing. We have implemented a prototype, and use it to showcase an example usage that performs system-level evaluation of the most popular AD AI attack category, camera-based STOP sign detection [157, 420], using 45 different combinations of system-level scenario setups (i.e., speeds, weather, and lighting). We find that the AI component-level and AD system-level results are quite different and often contradict each other in common driving scenarios, which further demonstrates the necessity and benefits of such system-level evaluation infrastructure building effort. Demos are available on our project website `https://sites.google.com/view/cav-sec/pass` [25].

In summary, this work makes the following contributions:

- We perform the first SoK of the growing semantic AD AI security research space. In total, we collect and analyze 53 such papers, and systematically taxonomize them based on research aspects critical for the security field, including the targeted AI component, attack/defense goal, attack vector, attack knowledge, defense deployability, defense robustness, and evaluation methodologies.
- We summarize 6 most substantial scientific gaps observed based on quantitative comparisons both vertically among existing AD AI security works and horizontally with security works from closely-related domains. With these, we are able to provide insights

and potential future directions not only at the design level, but also at the research goal, methodology, and community levels.

- To address the most critical scientific methodology-level gap, we take the initiative to develop an open-source, uniform, and extensible system-driven evaluation platform *PASS* for the semantic AD AI security research community. We also use our implemented prototype to showcase the capabilities and benefits of such a platform using representative AD AI attacks.

7.2 Systematization Scope

In our systematization, we consider within-scope the attacks that aim to address the *semantic AI security* challenges (defined in §7.1) and the defenses that are designed specifically for addressing the AI component-level vulnerabilities revealed in such attack works. Thus, we consider out-of-scope the works that assume digital space perturbations without justifying the feasibility in the AD context (i.e., without addressing the system-to-AI semantic gap) [301, 362, 179, 402] and the ones that focus only on AI algorithms that are not used in representative AD system designs today (§2.1.1), e.g., general image classification [228, 144, 118]. Curious readers can refer to general adversarial attack/defense SoK [300] and surveys [77, 114].

When collecting the papers, we mainly focus on the ones published in commonly-recognized top-tier venues [99] in closely-related fields to AD AI (i.e., security, Computer Vision (CV), Machine Learning (ML), AI, and robotics), as well as a few well-known works published in arXiv and other venues based on our best knowledge. Particularly, for the top-tier venues, we *exhaustively* search over the paper lists from 2017 to 2021 to find the ones that fall into our scope above.

7.3 Systematization of Knowledge

In this section, we taxonomize the semantic AD AI security works published in the past 5 years since the first one appeared in 2017. In total, 53 papers fall into our scope (§7.2) across security, CV, ML, AI, and robotics research areas; 48 discovered new attacks (Table 7.2) and 8 developed effective new defense solutions (Table 7.3). Fig. 2.9 shows the number of papers in each year. The earliest paper [255] dates back to 2017 when adversarial attacks (§2.1.2) began to be applied to many real-world application scenarios including AD. Since then, AD AI security has gained much attention as reflected in the exponential growth trend of paper numbers over the years.

Similar to many other fields, AD AI security research also has gone through enlightening periods where early beliefs are later overturned. For example, the 2017 paper [255] questions the severity of adversarial attacks in AD vehicles and claims that "no need to worry about adversarial examples in object detection in autonomous vehicles". Soon after that, two papers in 2018 propose successful attacks against camera object detection with physical-world attack demonstrations. As the community exponentially grew after 2019 and now at over 50 papers in this research space, we think now might be a good time to systematize the existing research efforts.

7.3.1 (Attack/Defense) Targeted AI Components

Status and trends. The targeted AI components in the existing works are summarized in Tables 7.2 and 7.3. As shown in the tables and Fig. 7.1, most (>86%) of the existing works target perception, while localization, chassis, and end-to-end driving are all less or equal to 6.2%. Among the perception works, the two most popular ones are camera (60.0%) and LiDAR (21.5%) perception. More detailed summary of their designs, applications in AD

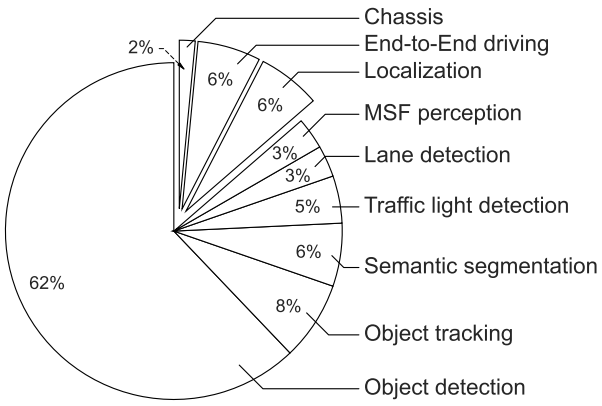

Figure 7.1: Distribution of (attack/defense) targeted AI components in semantic AD AI security papers.

systems, and vulnerabilities are in Appendix H. Currently, *none* of the existing works study the downstream AI components such as prediction and planning, which will be discussed more in §7.4.4.

7.3.2 Systematization of Semantic AD AI Attacks

Table 7.2 summarizes the semantic AD AI attacks. We taxonomize them based on 3 research aspects critical for the security field: Attack goal, attack vector, and attacker's knowledge.

Targeted AI component		Paper	Year	Field	Attack goal			Attack vector												Eval. level		
								Physical-layer										Cyber layer				
								Phys. world		Sensor attack												
					Integrity	Confidentiality	Availability	Object texture	Object shape	Object position	GPS spoofing	LiDAR spoofing	Radar spoofing	Laser/IR light	Acoustic signal	Translucent patch	ML backdoor	Malware & s/w comp.	Attacker's knowledge	Component-level	System-level	Open source
Camera perception	Object detection	Lu et al. [255]	'17	V	✓			✓											○	✓		
		Eykholt et al. [157]	'18	S	✓			✓											○	✓		
		Chen et al. [120]	'18	M	✓			✓											○	✓		✓
		Zhao et al. [420]	'19	S	✓			✓											○	✓		
		Xiao et al. [401]	'19	V	✓			✓	✓										○	✓		
		Zhang et al. [417]	'19	M	✓			✓											◒	✓		✓
		Nassi et al. [283]	'20	S	✓			✓											○	✓	✓	
		Man et al. [264]	'20	S	✓									✓					○	✓		✓
		Hong et al. [191]	'20	S	✓												✓		○		✓	
		Huang et al. [196]	'20	V	✓			✓											○	✓		✓
		Wu et al. [399]	'20	V	✓			✓											○	✓		
		Xu et al. [405]	'20	V	✓			✓											○	✓		
		Hu et al. [194]	'20	V	✓			✓											◒	✓		
		Hamdi et al. [181]	'20	M	✓				✓										◒	✓		
		Ji et al. [206]	'21	S	✓										✓				◒	✓		
		Lovisotto et al. [253]	'21	S	✓			✓											●	✓		✓
		Wang et al. [389]	'21	S	✓									✓					◒	✓		
		Köhler et al. [224]	'21	S	✓									✓					●	✓		
		Wang et al. [387]	'21	S	✓			✓											●	✓		✓
		Zolfi et al. [427]	'21	V	✓											✓			○	✓		
		Wang et al. [388]	'21	V	✓			✓											◒	✓		✓
		Zhu et al. [424]	'21	M	✓									✓					○	✓		
	Semantic segmentation	Nakka et al. [279]	'20	V	✓			✓											○	✓		
		Nesti et al. [285]	'22	V	✓			✓											○	✓		

		Jha et al. [205]	'20	S	✓				✓	○		✓	
	Object tracking	Jia et al. [209]	'20	M	✓	✓				○	✓		✓
		Ding et al. [143]	'21	M	✓	✓				○	✓		
		Chen et al. [121]	'21	M	✓	✓				○	✓		
	Lane detection	Sato et al. [327]	'21	S	✓	✓				○	✓	✓	✓
		Jing et al. [210]	'21	S	✓	✓				◐	✓	✓	
	Traffic light detection	Wang et al. [389]	'21	S	✓			✓		○	✓		
		Tang et al. [355]	'21	S	✓		✓			○		✓	
LiDAR perception	Object detection	Cao et al. [111]	'19	S	✓		✓			○	✓	✓	
		Sun et al. [347]	'20	S	✓		✓			●	✓		
		Hong et al. [191]	'20	S	✓				✓	○		✓	
		Tu et al. [371]	'20	V	✓	✓				○	✓		
		Zhu et al. [426]	'21	S	✓		✓			◐	✓		
		Yang et al. [410]	'21	S	✓	✓				◐	✓	✓	
		Hau et al. [184]	'21	S	✓		✓			○	✓		
		Li et al. [243]	'21	V	✓		✓			○	✓		✓
		Zhu et al. [425]	'21	O	✓			✓		◐	✓		
	Semantic segmentation	Tsai et al. [369]	'20	M	✓	✓				○	✓		✓
		Zhu et al. [425]	'21	O	✓			✓		◐	✓		
RADAR perception	Obj. detection	Sun et al. [349]	'21	S	✓			✓		◐	✓	✓	
MSF perception		Cao et al. [110]	'21	S	✓	✓				○	✓	✓	✓
		Tu et al. [370]	'21	O	✓	✓				○	✓		
LiDAR localization		Luo et al. [257]	'20	S	✓				✓	○		✓	
MSF localization		Shen et al. [335]	'20	S	✓		✓			○	✓	✓	
Camera localization		Wang et al. [389]	'21	S	✓			✓		○	✓		
Chassis		Hong et al. [191]	'20	S	✓				✓	○		✓	
End-to-end driving		Liu et al. [251]	'18	S	✓	✓			✓	○	✓	✓	✓
		Kong et al. [226]	'20	V	✓	✓				○	✓		✓
		Hamdi et al. [181]	'20	M	✓	✓				◐	✓		
		Boloor et al. [103]	'20	O	✓	✓				●	✓	✓	✓

Field: S = Security, V = Computer Vision, M = ML/AI, O = Others, e.g., Robotics, arXiv; s/w = software, comp = compromise

Attacker's knowledge: ○ = white-box, ◐ = gray-box, ● = black-box

Table 7.2: Summary of existing semantic AD AI attacks.

Attack Goals

We categorize the attack goals in the existing works based on the general security properties such as *integrity*, *confidentiality*, and *availability* [343]:

Integrity (of AI components). Integrity in AD context can be viewed as the integrity of the AI component outputs (i.e., whether they are changed by the attacker), which can directly impact the correctness of the AI driving behaviors. From this view, its violations manifest as the following AD-specific attack goals considered in existing works:

- *Safety hazards.* Safety is the top priority in AD design [85], and it is not only for the AD vehicle and its passengers, but also for other road users (e.g., other vehicles, pedestrians). Many existing attacks aim for safety hazards. For example, attacks on object detection and segmentation can cause safety hazards if a front vehicle is undetected [417, 279, 388]; attacks on object tracking can potentially lead to collisions if the front vehicle trajectory is incorrectly tracked [209, 205]; lane detection, localization, and end-to-end driving model attacks [327, 210, 335, 251, 226] can cause lane departure and thus potential collisions.

- *Traffic rule violations.* Since AD systems by design are required to follow traffic rules, violations can lead to financial penalties for individuals and reputational losses for AD companies. Existing attacks aim to hide the STOP sign to cause STOP sign violations [120, 157, 420]. Traffic light detection attack can cause the AD system to recognize a red light as green light, which may lead to red light violations [355]. In addition, the attacks on lane detection, localization, and end-to-end driving models can also cause vehicle lateral deviations, which can violate the lane line boundaries [327, 210, 335, 251, 226].

- *Mobility degradation.* A key benefit that AD technologies can bring is improved access to mobility-as-a-service (MaaS) [100]. A few works aim to degrade the mobility of

AD vehicles by manipulating the AI component outputs. In those works, they fool either the object detection to recognize a static blocking obstacle [111, 347, 410] or the traffic light detection to recognize a permanent red light [355], which can cause the AD vehicle to be stuck on the road for a prolonged time. This not only delays the AD vehicle's trip to destination, but also may block the traffic and cause congestion.

Confidentiality (Privacy). Confidentiality is related to the sensitive information from or collected by the AD vehicle. This includes not only the vehicle identification information (e.g., the VIN from Chassis [191]), but also the privacy-sensitive location data [257] as it can reveal the passenger privacy.

Availability (of AI components). Availability in the general cybersecurity area means the systems, services, and information under viewed are accessible to users when they need them [343]. For an AD AI component, its "users" can be considered as all the downstream AD components and vehicle subsystems that are counting on its timely and reliable outputs to function correctly. Thus, the availability of an AD AI component can be defined as its capability to provide timely and reliable outputs.

Following this definition, example attacks on AD AI availability can be attacks causing delays or failures in the outputting function of a given AI component, e.g., by interrupting its input/output messaging channels, causing software crashes/hangs in it, or by causing the vehicle system to fall back to human operators (so stop the outputting function of the whole AD AI stack).

Status and trends. Vast majority (96.3%) of existing works focus on integrity, while only 3.7% are on confidentiality and so far none of them are on availability. More discussion on this are in §7.4.5.

Attack Vectors

Existing attacks on AD systems leverage a diverse set of attack vectors and we broadly categorize them in two categories: *physical-layer* and *cyber-layer*. Physical-layer attack vectors refer to those tampering the sensor inputs to the AI components via physical means. We further decompose physical-layer attack vectors into *physical-world attack* and *sensor attack* vectors, where the former modifies the physical-world driving environment and the latter leverages unique sensor properties to inject erroneous measurements. Cyber-layer attack vector refers to those that require internal access to the AD system, its computation platform, or even the system development environment.

Physical-world attack vectors:

- *Object texture* refers to changing the surface texture (e.g., color) of 2D or 3D objects. It is often used by adversarial attacks to embed the malicious sensing inputs. In attack deployment, this is often fabricated as patches [327, 157, 420], posters [255, 120, 157], and camouflages [417, 196, 194], or displayed by projectors [253] and digital billboards [283]. Existing attacks have applied this attack vector on various objects such as STOP signs [255, 157, 120, 420], road surfaces [327, 210], vehicles [417, 196, 194, 388], clothes [399, 405], physical billboards [226]. To disguise as benign looking, a few attacks also constrain the texture perturbations to improve the attack stealthiness [196, 327, 210].

- *Object shape* refers to perturbing the shape of 3D objects such as vehicles [401, 369], traffic cones [110], rocks [110] or irregularly-shaped road objects [410, 371, 370]. Some were demonstrated in physical world via 3D printing [410, 111].

- *Object position* refers to placing physical objects at specific locations in the environment. Prior work [426] applies this attack vector by controlling a drone to hold a board at a particular location in the air to fool the LiDAR object detector to misdetect the

front vehicle.

Sensor attack vectors:

- *LiDAR spoofing* refers to injecting additional points in the LiDAR point cloud via laser shooting. Prior works carefully craft the injected point locations to fool the LiDAR object detection [111, 347, 184].

- *RADAR spoofing* refers to injecting malicious signals to RADAR inputs to cause it to resolve fake objects at specific distances and angles. Prior work [349] demonstrates RADAR spoofing capability in physical world and shows that it can cause AD system to recognize fake obstacles.

- *GPS spoofing* refers to sending fake satellite signals to the GPS receiver, causing it to resolve positions that are specified by the attacker. Prior works leverage GPS spoofing to attack MSF localization [335], LiDAR object detection [243], and traffic light detection [355].

- *Laser/IR light* refers to injecting/projecting laser or light directly to the sensor rather than the environment. Prior works use such attack vector to project malicious light spots in the camera image such that it can misguide the camera localization [389] or object detection [389, 424]. Moreover, some prior works also use it to cause camera effects such as lens flare [264] and rolling shutter effect [224] in order to fool the object detection.

- *Acoustic signal* has been shown to disrupt or control the outputs of Inertial Measurement Units (IMUs) [341, 368]. Prior work [206] used this attack vector to attack the camera stabilization system, which has built-in IMUs, to manipulate the camera object detection results.

- *Translucent patch* refer to sticking translucent films with color spots on camera lens.

It can cause misdetect road objects such as vehicles and STOP signs [427].

Cyber-layer attack vectors:

- *ML backdoor* is an attack that tampers the training data or training algorithm to allow the model to produce desired outputs when specific triggers are present. Prior work [251] leverages this to attack the end-to-end driving model by presenting the trigger on a roadside billboard.

- *Malware & software compromise* is generic cyber-layer attack vectors assumed in prior works to eavesdrop or modify sensor data [205], or execute malicious programs alongside the AI components [257, 191]. Particularly, a domain-specific instance is the Robot Operating System (ROS) node compromises [191], which can modify inter-component messages within ROS-based AD systems, e.g., Apollo v3.0 and earlier [7] and Autoware [213].

Status and trends. As shown in Table 7.2, existing works predominantly adopt physical-layer attack vectors, with 63.0% and 27.8% using physical-world and sensor attack vectors, respectively. Among the physical-world ones, object texture is the most popular (half of the all attacks). This is likely because such attack vectors are direct physical-world adaptations of the digital-space perturbations in general adversarial attacks. In contrast, only 6 (11.1%) attacks leverage cyber-layer attack vectors. More discussions on this are in §7.4.3.

Attacker's Knowledge

We follow the general definitions of attacker's knowledge by Abdullah et al. [73]:

White-box. This setting assumes that the attacker has complete knowledge of the AD system, including the design and implementation of the targeted AI components, corresponding sensor parameters, etc. Among existing attacks, such white-box setting is the

most commonly-adopted (Table 7.2).

Gray-box. This setting assumes that part of the information required by white-box attacks is unavailable. Prior gray-box attacks all assume the lack of knowledge of AI model internals such as weights. However, some works still require confidence scores in the model outputs [417, 194, 181, 206, 388, 210, 426, 410], and some require detailed sensor parameters [389, 349, 206].

Black-box. This is the most restrictive setting where the attacker cannot access any of the internals in the AD vehicle. Prior works that belong to this category are either transfer-based attacks [253, 387], which generate attack inputs based on local white-box models, or the ones that do not require any model-level knowledge in attack generation [347, 224].

Status and trends. As shown in Table 7.2, existing works commonly assume the white-box settings in their attack designs (66.7%). However, in recent years, we start to see increasing research efforts in the more challenging but also more practical attack settings, i.e., gray-box (24.1%) and black-box (9.3%), mostly published in recent two years (except one). This greatly benefited the practicality and realism of this research space, and also shows that the community practices have evolved significantly from earlier years [142, 312].

7.3.3 Systematization of AD AI Defenses

Table 7.3 summarizes the defenses included in our SoK. We taxonomize them from 4 research aspects critical for the security field: Defense methods, defense goals, defense deployability, and robustness to adaptive attacks.

Target AI Component		Paper	Year	Field	Defense method	Defense goal	Deployability: Neg. timing overhead	Deployability: Neg. resource overhead	Deployability: No model training	Deployability: No additional dataset	Deployability: No h/w modification	Robust to adaptive attack	Eval. level: Component-level	Eval. level: System-level	Open source
Camera perception	Object detection	Nassi et al. [283]	'20	S	Driving context & physics consistency checking	D	✓				✓	✓	✓		✓
		Li et al. [242]	'20	V	Driving context consistency checking	D	?				✓	?	✓		✓
		Liu et al. [248]	'21	S	Sensor fusion based consistency checking	D						✓	✓		
		Chen et al. [117]	'21	V	Adversarial training	M	✓	✓		✓	✓	?	✓		
	Obj. tracking	Jia et al. [207]	'20	V	Predict & remove perturbation from inputs	M	?				✓	?	✓		✓
Camera percep. & localization		Wang et al. [389]	'21	S	Consistency checking on reflected lights	D	?				✓	?	✓		
LiDAR percep.	Obj. detection	Sun et al. [347]	'20	S	Consistency checking on point cloud free space	D	✓	✓	✓	✓	✓	✓	✓		
		Sun et al. [347]	'20	S	Augment inputs with obj. conf. from front view	M	?			✓	✓	✓	✓		
		You et al. [412]	'21	S	Trajectory consistency checking	D	✓		✓	✓	✓	?	✓		

Field: S = Security, V = Computer Vision, M = ML/AI, O = Others, e.g., Robotics, arXiv; Defense goal: D = detection, M = mitigation, **?** = not available in the paper and we cannot conclude from the design, conf. = confidence, h/w = hardware

Table 7.3: Summary of existing semantic AD AI defenses.

Defense Methods

In general, the defense methods in existing works can be categorized into two categories:

Consistency checking. Consistency checking is a general attack detection technique that cross-checks the attacked information either with other (ideally independent) measurement sources, or with invariant properties of itself. For example, prior works use the detected objects from stereo cameras [248] and object trajectory from prediction models [412] to cross-check LiDAR object detection results. Li et al. [242] and Nassi et al. [283] created detection models to determine whether the current camera object detection results are consistent with the driving context. Some works also leverage physical invariant properties of the sensor source such as light reflection [283, 389] and LiDAR occlusion patterns [347] to detect AI attacks.

Adversarial robustness improvement. Several defenses try to improve the robustness of the AI component against attacks. For example, Chen et al. [117] applied adversarial training [175] to make the camera object detection model more robust. Jia et al. [207] improved the model robustness by predicting and removing potential adversarial perturbations from the model inputs. Sun et al. [347] augment the point cloud with point-wise confidence scores from a front view segmentation model to improve the robustness of LiDAR object detection.

Status and trends. As shown in Table 7.3, existing defenses share common general defense strategies, with 66.7% (6/9) based on consistency checking and 33.3% (3/9) based on adversarial robustness improvement. We discuss a few possible new directions later in §7.4.2.

Defense Goals

The defense methodologies in the above section can be naturally mapped to two defense goals: (1) **Detection:** All consistency checking based defenses are designed to detect the

attack attempts; and (2) **Mitigation:** Improving the adversarial robustness of models can generally reduce the attack success rate and raise the bar of the attackers. However, it is not designed to detect the attack, and also cannot fundamentally eliminate/prevent AI vulnerabilities.

Status and trends. Among the existing defenses, attack detection and mitigation are the main focus; so far, none of the works aims for other goals such as attack prevention. More discussions on this are in §7.4.2.

Defense Deployability

In this section, we list the defense design properties that are highly desired for the deployment in practical AD settings:

Negligible timing overhead. Timing overhead is one of the most important factors that may limit the deployability of defense for real-time systems such as AD systems. Among the existing defenses, 4 have negligible or no timing overhead by design or shown in evaluation [283, 117, 347, 412], 1 fails to catch up with the camera and LiDAR frame rate in evaluation [248]. For the remaining ones, we cannot conclude their timeliness from their papers (e.g., no timing overhead evaluation).

Negligible resource overhead. In practice, the hardware that runs the AD system may have limited computation resources (e.g., limited GPUs) due to budget concerns. Among the defenses, except the ones that do not require *extra* ML models [117, 347, 349], all others will impose additional demand on the computation resources in order to execute the models. This may prevent them from deploying onto vehicles that have already fully utilized the hardware resources.

No model training. The requirement of model training imposes extra deployment burdens.

Among the defenses that require extra ML models, only one [412] uses public pre-trained models without fine-tuning.

No additional dataset. Closely related to the previous property, some defenses not only require model training, but also need additional datasets during training process. This will limit the deployability due to the efforts in dataset preparation.

No hardware modification. This property means whether the defense requires changes to the hardware on AD vehicle or adding new hardware (e.g., additional sensors). Among the defenses, Liu et al. [248] require stereo cameras, which may not be available on some AD vehicles.

Status and trends. As shown in Table 7.3, almost all existing defenses (7/8) have the awareness for at least one of these deployability design aspects, especially for that regarding hardware modification (7/8) and timing overhead (4/8). However, the awareness on the need for no model training, negligible resource overhead, and no additional dataset are currently lacking (2/8, 2/8, and 3/8 respectively).

Robustness to Adaptive Attacks

Adaptive attacks are designed to circumvent a particular defense. They often assume complete knowledge of the defense internals, including the design and implementation, and are designed to challenge the fundamental assumptions of the defense. Adaptive attack evaluation has become *strongly-advocated* in recent adversarial AI defenses, and prior work has also proposed guidelines on how to properly design adaptive attacks [366]. However, among the AD AI defenses, we find that *only 3 papers* conduct some forms of adaptive attack evaluation [283, 248, 347]. Specifically, Nassi et al. [283] applied existing adversarial attacks on their defense model and find that it is challenging to circumvent the attack detection; Liu et al. [248] evaluated simultaneous attacks against their consistency checking design between

LiDAR and camera object detection, and find that the detection is still effective. Sun et al. [347] designed adaptive attacks based on the defense assumptions (the LiDAR occlusion patterns), and show that they fail to break the two defenses.

Status and trends. Despite being strongly-advocated in general adversarial AI defense research [366], currently not many (3/8) of the existing defenses in AD AI security conduct evaluation against adaptive attacks. However, with the increasing adoption of such practices in the general adversarial AI domain, this situation on the semantic AD AI security domain should also see improvements.

7.3.4 Systematization of Evaluation Methodology

At the evaluation methodology level, besides the expected differences due to problem formulations (e.g., attack targets and goals), we notice an interesting general disparity in the choices of the *evaluation levels*, which is relatively unique for semantic AI security as opposed to generic AI security. Specifically, since here the attack targets are AI components but the ultimate goals are to achieve system-level attack effect (e.g., crashes), the evaluation methodology can be designed at both AI component and AD system levels:

Component-level evaluation is to evaluate attack/defense impacts only at the targeted AI component level (e.g., object detection) without involving (1) other necessary components in the full-stack AD systems (e.g., object tracking, prediction, planning); and (2) the closed-loop control [165]. The evaluation metrics are thus component-specific, e.g., detection rate [157, 420], location deviation [335], steering error [226], etc.

System-level evaluation refers to evaluating the attack/defense impacts at the vehicle driving behavior level considering full-stack AD system pipelines and closed-loop control. The evaluation setups used to achieve this can be generally classified into two categories: (1)

Real vehicle-based, where a real vehicle is under partial (e.g., steering only [210]) or full (i.e., steering, throttle, and braking [283, 349]) closed-loop control of the AD system to perform certain driving maneuvers with the presence of attacks, which are typically deployed in the physical world as well; and (2) *Simulation-based*, where the critical elements in the closed-loop control (e.g., sensing/actuation hardware and the physical world) are fully or partially simulated, using either existing simulation software [327, 355, 110, 335, 251] or custom-built modeling [335].

Status and trends. As shown in Table 7.2 and 7.3, the majority (90.5%) of the surveyed works performed component-level evaluation, likely due to the ease of experiment efforts (no need to set up real vehicle or simulation), while only 25.4% (16/63) adopted some forms of system-level evaluation. More discussions are in §7.4.1.

7.4 Scientific Gaps and Future Directions

Based on the systematization of existing works on AD AI security, we summarize a list of scientific gaps that we observed and also discuss possible solution directions. To avoid subjective opinions and bias, such observations are drawn from quantitative comparisons *vertically* among the choices made in existing AD AI security works along with different design angles in §7.3 and/or *horizontally* with security works in closely-related CPS (Cyber-Physical Systems) domains such as drone and Automatic Speech Recognition and Speaker Identification (ASR/SI) based on recent SoKs [282, 73].

7.4.1 Evaluation: General Lack of System-Level Evaluation

Scientific Gap 1: *It is widely recognized that in AD system, AI component-level errors do not necessarily lead to system-level effect (e.g., vehicle collisions). However, system-level*

evaluation is generally lacking in existing works.

As identified in §7.3.4, currently it is not a common practice for AD AI security works to perform system-level evaluation: overall only 25.4% of existing works perform that, and such a number is especially low (*7.4%*) for the most extensively-studied AI component, camera object detection. In fact, the vast majority (74.6%) of these works *only* performed component-level evaluation without making any efforts to experimentally understand the system-level impact of their attack/defense designs. Admittedly, for CPS systems such system-level evaluation is generally more difficult due to the involvement of physical components. However, we find that for existing security research on related CPS domains such as drone and ASR/SI, actually *almost all* attack works perform system-level evaluations (100% for drone, 94% for ASR/SI based on the SoKs [282, 73]). This shows that the current AD AI security research community is particularly lagging behind in such common evaluation practice in CPS security research.

In the CPS verification area, it is actually already widely-recognized that for CPS with AI components, *AI component-level errors do not necessarily lead to system-level effects* [149, 332]. For AD systems, this is especially true due to the high end-to-end system-level complexity and closed-loop control dynamics, which can explicitly or implicitly create fault-tolerant effects for AI component-level errors. In fact, various such counterexamples have already been discovered in AD system context, e.g., when the object detection model error is at a far distance for automatic emergency braking systems [149, 332], or such errors can be effectively tolerated by downstream AI modules such as object tracking [209]. This means that even with high attack success rates shown at the AI component level, it is actually possible that such an attack *cannot cause any meaningful effect to the AD vehicle driving behavior*. For example, as concretely estimated by Jia et al. [209], for camera object detection-only AI attacks, a component-level success rate of up to 98% can still be not enough to affect object tracking results. Thus, we believe that for semantic AD AI security research, the current general lack of system-level evaluation is a critical *scientific methodology-level gap*

that should be addressed as soon as possible.

Future directions. Specific to the AD problem domain, there are various technical challenges to effectively fill this gap at the research community level. Among the possible options (§7.3.4), real vehicle-based system-level evaluation is generally unaffordable for most academic research groups (e.g., $250K per vehicle [133]), not to mention the need for easily-accessible testing facilities, the high time and engineering efforts needed to set up the testing environment, and also high safety risks (especially since most AD AI attacks aim at causing safety damages). Simulation-based evaluation is much more affordable, accessible, and safe, but researchers still need to spend substantial engineering efforts to instrument existing AD simulation setup for the security-specific research needs, e.g., modifying the simulated physical environment rendering [327, 110] or sensing channels [335] for launching attacks. These might explain why system-level evaluation is not commonly adopted for AD AI security research today.

To address such impasse, it is highly desired if there can be a community-level effort to collectively build a common system-level evaluation infrastructure, since (1) the engineering efforts spent in instrumenting either a real AD vehicle or AD simulation for security research evaluation share common design/implementation patterns (e.g., common attack entry points as in §7.3.2); and (2) in AD context, the system-level attack effect can be highly influenced by driving scenario setups (e.g., large braking distance differences in highway and local roads [55] and thus the system-level evaluation results are only comparable (and thus scientifically-meaningful) if the same evaluation scenario and metric calculation are used. Considering the criticality of such a methodology-level scientific gap, later in §7.5 we take the initiative to lay the groundwork to foster such a community-level effort.

7.4.2 Research Goal: General Lack of Defense Solutions

Scientific Gap 2: *Compared to AD AI security attacks, their effective defense solutions are substantially lacking today, especially those on attack prevention.*

As shown in Tables 7.2 and 7.3, among existing semantic AD AI security works, the ratio of discovered attacks (85.7%) is much higher than effective defense solutions (14.3%). Furthermore, all existing defenses focus on attack detection and mitigation, and *none* studied *prevention*, which is a more ideal form of defense. In comparison, such ratios are much more balanced in the drone security field [282], where 51% are on attacks and 49% are on defenses. Also, among the defenses, *33%* are for prevention. While attack discovery is the necessary first step for security research into an emerging area, it is imperative to follow up with effective defense solutions, especially those on attack prevention, to close the loop of the discovered attacks and actually benefit the society.

Future directions. As shown in Tables 7.2 and 7.3, many of the AD AI components with discovered attacks actually do not have any effective defense solutions yet (e.g., lane detection, MSF perception), which are thus concrete defense research directions. Note that there indeed exist generic AI defenses that are directly applicable (e.g., input transformation [407]), but it is already found that such generic defenses are generally ineffective against the CPS domain AI attacks, e.g., for both AD [327, 110] and ASR/SI [199].

Besides considering the currently-undefended AI components, there are also possibly-applicable defense strategies that have not been explored yet, for example certified robustness [232]. While unexplored (§7.3.3), such a defense is actually highly desired in the AD AI security domain since it can provide strong theoretical guarantees for AI security in such a safety-critical application domain. The challenge is that today's certified robustness designs only focus on small 2D digital-space perturbations, and thus their extensions to today's popular AD AI attack vectors (e.g., physical-world or sensor attacks) are still open research problems. This requires addressing at least the system-to-AI semantic gap, not to mention potential deployability challenges as such methods generally have high overhead [400] and thus might

be hard to meet the strong real-time requirement in AD context (§7.3.3).

7.4.3 Attack Vector: Cyber-Layer Attack Vectors Under-Explored

Scientific Gap 3: *Existing works predominately focus on physical-layer attack vectors, which leaves the cyber-layer ones substantially under-explored.*

As shown in Table 7.2, there are only 11.1% (6/54) AD AI attack works that leverage cyber-layer attack vectors in their attack designs, assuming ML backdoors [251], malware [205], remote exploitation [257], and compromised ROS nodes [191], respectively. Among these four, only the last one is relatively domain-specific to AD. In related CPS domains such as drones and ASR/SI, such ratio is much higher (>50%): 53.8% for drone attack works [282] and 52.9% for ASR/SI ones [73]. One observation we have is that many of these works in related CPS domains exploit domain-specific networking channels such as ground station communication channel [256] and First-Person View (FPV) channel [323, 276, 141] in drones, audio files, and telephone networks [72] in ASR/SI systems. However, *no existing works in AD AI security* studied such aspects, which can thus be one direction to fill this research gap.

Future directions. The general design goal of AD AI stack is indeed mostly towards achieving autonomy only using on-board sensing; however, this does not mean that there are no networking channels that can severely impact end-to-end driving. For example, at least the following two are domain-specific, widely-adopted in real-world AD vehicles, and highly security and safety critical:

High Definition (HD) Map update channel. For L4 vehicles, the accuracy of HD Map is critical for driving safety as it is the fundamental pillar for almost all AD modules such as localization, prediction, routing, and planning. Real world dynamic factors (e.g., road constructions), make it imperative to keep the HD Map frequently updated, which is typically over-the-air. For example, when a Waymo AD vehicle detects a road construction, it updates

the HD Map and shares the map update with the operations center and the rest of the fleet in real time [393].

Remote AD operator control channel. Another channel that is also relevant to high-level AD is the control channel for remote operators. Unlike lower-level AD vehicles, L4 and above do not have safety drivers onboard. However, after an AD vehicle reaches a fail-safe state, a remote operator usually take over the control (e.g., happened to a Waymo vehicle stuck on the road [394]), which is directly safety-critical if hijacked.

7.4.4 Attack Target: Downstream AI Component Under-Explored

Scientific Gap 4: *There is a substantial lack of works focusing on downstream AI components (e.g., prediction, planning), which are equally (if not more) important compared to upstream ones w.r.t system-level attack effect.*

Among the vast majority (50/54) of attack works that target the more practical modular AD system designs (§2.1.1), so far *none* of them targeted downstream AI components such as prediction and planning; they predominately focus on the upstream ones such as perception and localization. This is understandable since under the most popularly-considered physical-layer attack model (§7.3.2), upstream ones can be much more directly influenced by attack inputs (e.g., via physical-world or sensor attacks), while to affect the inputs of downstream ones such as predication, one has to manipulate the upstream component outputs (e.g., object detection) first. However, these downstream ones are actually at least equally (if not arguably more) important than the upstream ones. For example, errors in the obstacle trajectory prediction or path planning will actually more directly affect driving decisions and thus lead to system-level effects. Thus, it is highly desired for future AD AI security research to fill this gap.

Future directions. To study downstream AI component security, a general challenge is how to study their component-specific security properties with sufficient isolation with upstream

component vulnerabilities, so that we can isolate the causes and gain more security design insights specific to the targeted downstream component. Here we discuss several possible solution directions:

Physical-layer attacks by road object manipulation. The prediction component in AD systems predicts obstacle trajectory based on its detected physical properties (e.g., obstacle type, dimension, position, orientation, speed). Therefore, assuming upstream components such as AD perception is functioning correctly, the attacker can directly control a road object in the physical world (e.g., an attack vehicle on the road), in order to control the inputs of prediction without relying on vulnerabilities in upstream components. However, this requires the attack design to ensure that the triggers of the prediction vulnerability are *semantically-realizable*, e.g., the attacker-controlled obstacle needs to have a consistent type, no breaks in the historical trajectory, moving at a reasonable speed, etc.

Physical-layer attacks by localization manipulation. The planning component takes the prediction and localization outputs as inputs to calculate the optimal driving path in the near future. Therefore, the change in the localization directly affects the decision-making in the planning. To attack the planning, one can leverage localization-related attack vectors (e.g., GPS spoofing) to control planning inputs. However, this only works for AD systems that purely rely on such input (e.g., GPS) for localization; if MSF-based localization is used, it is still hard to isolate planning-specific vulnerabilities from MSF algorithm vulnerabilities (e.g., [335]).

Cyber-layer attacks. Perhaps the most direct way to manipulate the inputs to the downstream components is via cyber-layer attacks. For example, a compromised ROS node [191] in the AD system can directly send malicious messages to prediction or planning. Unlike the road object manipulation method, this does not require the inputs to be semantically-realizable. This thus forms another motivation to explore more on cyber-layer attacks (§7.4.3).

7.4.5 Attack Goal: Goals Other Than Integrity Under-Explored

Scientific Gap 5: *Existing attacks predominantly focus on safety/traffic rule violations or mobility degradation (can be viewed as integrity in AD domain), while other important security properties such as confidentiality/privacy, availability, and authenticity are under-explored.*

As shown in §7.3.2, almost all (96.3%) existing attacks target the AD-specific manifestation of integrity (i.e., modify the driving correctness). However, the study of other important security properties such as confidentiality and availability [343] is substantially lacking: only 2 (3.7%) touch upon confidentiality and none of them study availability. In comparison, the attack goals in drone security are more balanced: 57.9% (11/19) on integrity/safety, 31.6% (6/19) on privacy/confidentiality, and 88.9% (8/19) on availability. Although integrity is highly important in AD context as vehicles are heavy and fast-moving, these other properties are also equally important from the general security field perspective.

Future directions. To foster future research into these less-explored security properties, here we discuss a few possible new research angles. For confidentiality/privacy, existing efforts only consider AD system-internal information leakage (e.g., location), but one can also consider potential private information extraction from AD system-*external* information such as from sensor inputs. In the drone security area, one main privacy attack scenario is along this direction, e.g., people-spying using the drone camera [390, 281]. For availability, since it is closely related to the communication channels among AI components, more extensive exploration on the cyber-layer attacks (§7.4.3) may make such attacks feasible. So far, no existing works consider authenticity in AD context, but in today's AD deployment and commercialization, authenticity can manifest as the authentication of the safety drivers, mutual authentication between the passenger and the AD vehicle for robotaxi, and mutual authentication between the consumers and the delivery vehicle for AD goods delivery, which are within the broader scope of AD AI stack.

7.4.6 Community: Substantial Lack of Open Sourcing

Scientific Gap 6: *The open-sourcing status of AD AI security works from the security community is much lacking compared to related communities/domains (e.g., CV, ML/AI, ASR/SI), which harms the reproducibility and comparability for this emerging area in the security community.*

For each work in Table 7.2 and Table 7.3, we searched for their code or open implementation in the paper and also on the Internet. Overall, there are less than *20.6% (7/34)* papers from security conferences that release source code. The situation is much worse if we narrow it down to the sensor attack works, where only *1 (8.3%) out of 12* works released its attack code. In comparison, the papers published in CV and ML/AI conferences have over 50% open-source percentages. Interestingly, also in the CPS domain, 50% ASR/SI papers in the security conferences release their code, which is roughly the same as in CV and ML/AI conferences. This indicates that it is not that security researchers are not willing to share code, but more likely related to the *diversity* of AD AI security papers in the security conferences. In fact, security conference papers adopt a much more diverse set of attack vectors than any other conferences as shown in Table 7.2. For example, among the 15 works that use sensor attack vectors, only 2 are from CV conferences and 1 is from ML/AI conferences. For those papers, it is indeed more difficult to share the code or implementation since hardware designs are usually involved. In comparison, the attack vectors in ASR/SI are much more uniform; they predominantly leverage malicious sound waves, which is convenient to modify, and can be evaluated digitally.

Future directions. There is no doubt that we should encourage more open-sourcing efforts from the security community such that future works can benefit from it. However, the challenge is in which form the hardware implementations should be shared such that such sharing can be more directly useful for the community. Here we discuss two possible solution directions based on our observations from prior works:

Open-source hardware implementation references. Since the reproduction cost and effort for hardware implementations are generally higher than software-only ones, it is actually more desired for researchers to release as many details about their hardware design as possible. This often means the circuit diagram, printed circuit board layout, bill of materials, and detailed experiment procedures. One example is the LiDAR spoofing work by Shin et al. [336], where they clearly list such details on a website and thus other groups were able to reproduce and build upon their designs [111].

Open-source attack modeling code. Another interesting trend in recent sensor attack works is that they often model the attack capability in the digital space for large-scale evaluation. For example, Man et al. [264] modeled the camera lens flare effect caused by attacker's light beams in digital images, Ji et al. [206] modeled the image blurry effect caused by adversarial acoustic signals on the camera stabilization system; Cao et al. [111] models the LiDAR spoofing capability as the number of points that can be injected to the point cloud; Shen et al. [335] models GPS spoofing as arbitrary GPS position that can be controlled by the attacker. Since these modelings are either validated in the physical-world experiments or backed up by the literature, sharing such code will greatly ease the reproduction in future works.

Our initiated community-level evaluation infrastructure development effort in §7.5 may also help fill this gap by encouraging open-sourcing practices (§7.5.1).

7.5 *PASS*: System-Driven Evaluation Platform for AD AI Security

Among all the identified scientific gaps, the one on the general lack of system-level evaluation (§7.4.1) is especially critical as proper evaluation methodology is crucial for valid scientific

progress. In this section, we take the initiative to address this critical scientific methodology-level gap by developing an open-source, uniform, and extensible system-driven evaluation platform, named *PASS* (Platform for Autonomous driving Safety and Security).

7.5.1 Design Goals and Choices

As described in §7.4.1, to effectively fill this gap at the research community level, a common system-level evaluation infrastructure is highly desired. To foster this, we take the initiative to build a *system-driven evaluation platform* that unifies the common system-level instrumentations needed for AD AI security evaluation based on our SoK (§7.3), and abstracts out the customized attack/defense implementation and experimentation needs as platform APIs. Such a platform thus directly provides the aforementioned evaluation infrastructure as it allows semantic AD AI security works to directly plug in their attack/defense designs to obtain system-level evaluation results, such that (1) without the need to spend time and efforts to build the lower-level evaluation infrastructure; and (2) generating results in the same evaluation environment and thus directly comparable to prior works. Note that although some existing works developed simulation-based system-level testing methods for AI-based AD [149, 293, 373, 150], their designs are for safety not security (no threat models), and thus cannot readily support the evaluation of different AD AI attack/defense methods.

This platform will be fully open-sourced so that researchers can collectively develop new APIs to fit future attack/defense evaluation needs, and also contribute attack and defense implementations to form a semantic AD AI security benchmark, which can improve comparability, reproducibility, and also encourage open-sourcing (currently lacking in the security community as in §7.4.6). We also plan to provide remote evaluation services for research groups that do not have the hardware resources required to run this platform, which will be maintained by the project team.

Method	FD	AF	SF	FX	EF	RP	Papers
Real vehicle-based	●	○	○	○	○	○	[283, 210, 349]
Simulation-based	◐	●	●	●	●	●	[205, 327, 355, 410, 110, 257, 335, 251]

FD = fidelity, AF = affordability, SF = safety, FX = flexibility,
EF = efficiency, RP = reproducibility; ○ = low, ◐ = medium, ● = high

Table 7.4: System-level evaluation methods used in existing works. Such complementariness motivates us to choose a *simulation-centric hybrid design* for our evaluation platform.

Simulation-centric hybrid design. To build such a community-level evaluation infrastructure, a fundamental design challenge is the trade-off between real vehicle-based and simulation-based evaluation methodology, which is shown in Table 7.4. Real vehicle-based is more fidel as it has the vehicle, sensors, and physical environment all in the evaluation loop, but simulation is better in all other important research evaluation aspects, ranging from cost, free of safety issues, high flexibility in scenario customization, convenience of attack deployment, much faster evaluation iterations, to high reproducibility. Considering their strengths and weakness, we choose a *simulation-centric hybrid design*, in which we mainly design for simulation-based evaluation infrastructure and only use real vehicles for simulation fidelity improvement (e.g., digital twin [351] and sensor model calibration [351, 265, 278, 396, 123]) and exceptional cases where it is easy to set up the physical scenario and maintain safety.

We choose the design to be simulation-centric because we find that the fidelity drawback of simulation-based approach is tolerable since (1) our results show that for today's representative AD AI attacks, the attack results and characteristics are highly correlated in today's industry-grade simulator and physical world, which indicates sufficient fidelity at least for such AD AI security evaluation purpose (detailed in §7.5); and (2) the simulation fidelity technology is still evolving as this is also the need for the entire AD industry [136, 395], for example recently there are various new advances in both industry [351] and academia [351, 265, 278, 396, 123]); and (3) after all the simulation environment can be considered as a different environmental domain, and practical attacks/defenses should be capable of such domain adaptations. Such a design is also more beneficial from the commu-

nity perspective as it makes the setup of such a platform more affordable for more research groups.

7.5.2 PASS Design

Our system-driven evaluation platform *PASS* is guided by two general design rationales: (1) *uniform*, aiming at unifying the common system-level evaluation needs at attack/defense implementation, scenario setup, and evaluation metric levels observed in our SoK (§7.3); and (2) *extensible*, aiming at making the platform capable of conveniently supporting new future attacks/defenses, AD system designs, and evaluation setups. Specifically, for attacks/defenses on a certain driving task, our platform defines a list of driving scenarios and metrics, which help unify the experimental settings and effectiveness measurements. To enable extensibility for future research, the platform provides a set of interfaces to simplify the deployment of new attacks/defenses. Based on our SoK, we design the attack/defense interfaces to support the common categories of attack vectors (§7.3.2) and defense strategies (§7.3.3). In addition, the platform provides a modular AD system pipeline with pluggable AI components, which makes the platform easily extensible to different AD system designs.

Fig. 7.2 shows the *PASS* design, consists of 6 main modules:

AD model. The AD model hosts the end-to-end AD system with the targeted AI components under testing. Specifically, the platform provides two AD model choices: modular AD pipeline and full-stack AD system (e.g., OpenPilot [294], Apollo [7], and Autoware [213]). The modular AD pipeline is provided as a flexible option to enable AD systems with replaceable AI components and different system structures. To configure the modular AD pipeline, the user only needs to modify a human-readable configuration file to specify the desired AI components to be included.

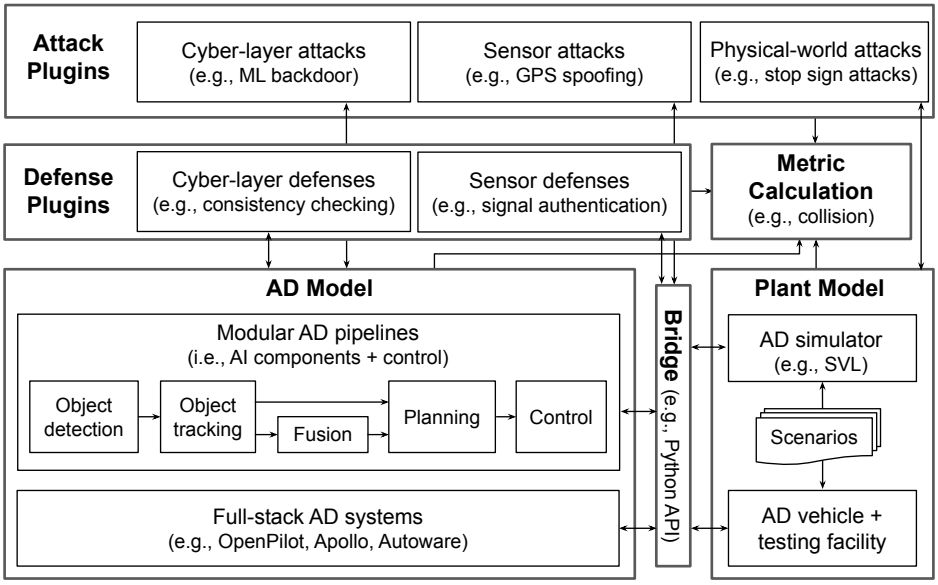

Figure 7.2: Design of our system-driven evaluation platform *PASS* for the semantic AD AI security community.

Plant model. This is the testing vehicle and physical driving environment. As described in §7.5.1, our platform is mainly built upon AD simulation (e.g., industry-grade ones such as SVL [238]). To configure the simulation environment, we define a set of scenarios to describe the AD vehicle initial position, equipped sensors, testing road, and road objects (e.g., vehicles and pedestrians). The scenario descriptions are also provided as human-readable files for easy modification. Benefited from the general AD model structure, the evaluation can also be performed on a real AD vehicle (e.g., L4-capable vehicles as shown in Fig. 7.3) under the driving scenarios provided by the testing facility.

Bridge. Bridge is the communication channel between the AD and plant models for sensor data reading and AD vehicle actuation. For better extensibility, it supports function hooking for modifying the communication data at runtime.

Attack/Defense plugins. These two host the available attacks and defenses, which are built upon the attack/defense interfaces provided by the platform. To ease the deployment of future attacks/defenses, these interfaces are designed to support common attack vectors (§7.3.2) and defense strategies (§7.3.3). Specifically, 3 general types of interfaces are defined

Figure 7.3: AD vehicles for the system-driven evaluation platform: (a) A real-vehicle sized chassis with L4 AD sensors and closed-loop control; (b) An L4 AD vehicle built upon Lincoln MKZ. Both are equipped with L4 AD-grade sensors such as LiDARs, cameras, RADARs, GPS, and IMU.

based on the SoK:

- *Physical-world attack interface* enables dynamically loading 2D/3D objects to the simulation environment at arbitrary locations. It covers many physical-world attacks leveraging object texture, shape, and position.
- *Sensor attack/defense interface* enables registering customized sensor data operations, which model the sensor attack or defense behaviors, in the bridge.
- *Cyber-layer attack/defense interfaces* support adding/replacing components in the AD system and serve as versatile interfaces to enable cyber-layer attacks/defenses. For example, ML backdoor attacks can be implemented by replacing an existing ML model with a trojan one. Consistency-checking based defense can be implemented by adding component with consistency checking logic.

Metric calculation. This is in charge of collecting measurements from all other modules in the platform and calculating the scenario-dependent evaluation metrics. Based on our SoK of attack goals (§7.3.2), we include system-level metrics such as safety (e.g., collision rate [205, 327]), traffic rule violation (e.g., lane departure rate [327, 335]), trip delay, etc. For

comprehensiveness we also include component-level metrics (e.g., frame-wise attack success rate [420, 157, 120]).

Implementation. We have implemented several variations for each module in the platform. The AD model supports modular AD pipelines for STOP sign attack evaluation (Appendix J) and full-stack AD systems (Apollo [7] and Autoware [213]). Our plant model includes an industry-grade AD simulator [238] and real L4-capable AD vehicles (Fig. 7.3). For bridge, we reuse the ones (Python APIs, CyberRT, and ROS) provided by the AD simulator. Specifically, we modified the AD simulator, bridge, and modular AD pipeline to enable the 3 general attack/defense interfaces mentioned above. For metrics, we implemented collision and STOP sign-related violation checkings. Note that we *do not intend to cover all existing attacks/defenses, and also believe we should not*; our goal (§7.5.1) is to initiate a community-level effort to *collaboratively* build a common system-level evaluation infrastructure, which is the only way to make such community-level infrastructure support sustainable and up-to-date. We will fully open-source the platform (will be available on our project website [25]) and welcome community contributions of new attack/defense interfaces and implementations.

7.5.3 Case Study: System-Level Evaluation on Stop Sign Attacks

Evaluation methodology. In this section, we use *PASS* to perform system-level evaluations for the existing STOP sign attacks to showcase the platform capabilities and benefits. We choose STOP sign attacks because they are the most studied AD AI attack type in the literature (§5) and *none* of them have conducted system-level evaluations (§7.3.4).

Evaluated attacks. We evaluate the most safety-critical STOP sign AI attack goal: STOP sign *hiding*, which generally leverage malicious object textures to make a STOP sign undetected (§7.3.2). We reproduced 3 representative designs: *ShapeShifter (SS)* [120], *Robust Physical*

Figure 7.4: Reproduced semantic AD AI attacks on STOP sign hiding: (a) RP2 [157] in simulator, (b) SS [120] in simulator, and (c) SS [120] in physical world.

Perturbations (RP2) [157], and *Seeing Isn't Believing (SIB)* [420]. Fig. 7.4 (a) and (b) show the reproduced adversarial STOP signs. Details are in Appendix I.

AD model configurations. We configure the modular AD pipeline (§7.5.2) in the platform as a general AD system that includes camera object detection, object tracking, fusion (optional), planning, and control. Specifically, we configure 3 variations of the AD pipeline based on the availability of map and fusion (detailed configurations are in Appendix J).

Driving scenarios. We select a straight urban road with a STOP sign at an intersection shown in Fig. 7.4 (a) and (b). We evaluate under 5 common local-road driving speeds (10–30 mph with 5-mph step), 3 lighting conditions (sunrise, noon, sunset), and 3 weather conditions (sunny, cloudy, rainy). During evaluation, the malicious STOP sign images are deployed in the simulator via physical-world attack interface (§7.5.2).

Evaluation metrics. To evaluate system-level attack effectiveness, we select traffic rule related metrics for STOP sign scenarios: *stopping rate* refers to the percentage of successfully stopping (but may already violate the stop line), *violation rate* is the percentage of violating the stop line, *violation distance* is the distance over the stop line if violated. For comparison, we also calculate component-level metrics used in prior works [157, 420]: *frame-wise attack success rate*, f_{succ}, is the percentage of frames that are misdetected in different STOP sign distance ranges, *the best attack success rate in n consecutive frames*, $f_{\text{succ}}^{\max(n)}$, is proposed by Zhao et al. [420] as a better metric to measure whether enough success has been accumu-

lated. We use $n = 50$ (instead of 100 [420]) since most of our simulation scenarios complete within 100 frames due to the low driving speeds (i.e., short braking distances).

Result highlights. Our evaluation results show that at the component level, all 3 attacks have very high effectiveness: SS achieves $f_{\text{succ}} > 90\%$ in 10 m and $f_{\text{succ}}^{\max(50)}$ close to *100%* in all scenarios; RP2 and SIB both achieve $f_{\text{succ}}^{\max(50)} = 100\%$ when the speed is 10 mph. Both the aggregated attack results and those at specific distance/angle ranges are all very similar to the reported component-level attack effectiveness in the original papers (Appendix I). However, similar to prior security/robustness studies for other AD AI components [332, 209], we find that such high attack success is *far from achieving the system-level attack goals*: RP2 and SIB are not able to cause STOP sign violations at all across all driving scenario combinations; SS can only succeed when the speed is very low (10 mph) while failing for all other speeds (15-30 mph), which are more common speed ranges for STOP sign roads. Detailed results are in Table 7.5.

Such low system-level attack effectiveness is mainly due to the tracking component in the modular AD pipeline, which were set up using representative parameters recommended by Zhu et al. [423]. Although the attacks are quite effective at close distances, the STOP sign track created before is still alive, so that the AD system is always aware of the STOP sign and keeps committing the stop decisions. For SS with low speed (10 mph), the attack can hide the detection in more consecutive frames, and thus the STOP sign track can be eventually deleted. This thus again validates the non-triviality of the AI-to-system semantic gap, which further demonstrates the necessity of system-level evaluations for semantic AD AI security (§7.4.1). Meanwhile, here we are able to experimentally quantify whether and how much the system-level attack effectiveness for a given AD AI attack depends on the driving scenario (i.e., driving speed), which is only made possible with our simulation-centric design that can more flexibly support different driving scenarios, AD designs, and system-level metrics.

Attack	AD Pip	Speed (mph)	Lighting	Weather	Component-level metrics				System-level metrics		
					f_{succ}			$f_{\text{succ}}^{\max(50)}$	Stop rate	Violation rate	Violation distance
					<10m	<20m	<30m				
SS [120]	Map	10	Noon	Sunny	91.3%	78.8%	67.4%	100.0%	**0.0%**	**100.0%**	∞
		15			99.8%	82.2%	60.4%	99.8%	100.0%	0.0%	0 m
		20			100.0%	87.8%	68.6%	100.0%	100.0%	0.0%	0 m
		25			100.0%	88.0%	69.2%	100.0%	100.0%	0.0%	0 m
		30			100.0%	88.7%	71.2%	98.0%	100.0%	0.0%	0 m
		25	Noon	Cloudy	100.0%	88.8%	69.8%	100.0%	100.0%	0.0%	0 m
				Rainy	99.3%	86.7%	68.3%	99.0%	100.0%	0.0%	0 m
			Sunrise Sunset	Sunny	100.0%	83.9%	66.0%	100.0%	100.0%	0.0%	0 m
					100.0%	88.0%	69.2%	100.0%	100.0%	0.0%	0 m
	Pinhole	10	Noon	Sunny	91.3%	78.8%	67.5%	100.0%	**0.0%**	**100.0%**	∞
		15			100.0%	82.5%	60.6%	100.0%	100.0%	0.0%	0 m
		20			100.0%	87.8%	68.6%	100.0%	100.0%	0.0%	0 m
		25			100.0%	88.5%	69.5%	100.0%	100.0%	0.0%	0 m
		30			100.0%	85.4%	68.6%	97.1%	100.0%	0.0%	0 m
	Fusion	10	Noon	Sunny	100.0%	86.1%	73.8%	100.0%	100.0%	0.0%	0 m
		15			100.0%	82.3%	60.4%	100.0%	100.0%	0.0%	0 m
		20			100.0%	87.8%	68.6%	100.0%	100.0%	0.0%	0 m
		25			100.0%	89.8%	70.6%	99.8%	100.0%	0.0%	0 m
		30			100.0%	88.7%	71.2%	97.5%	100.0%	0.0%	0 m
RP2 [157]	Map	10	Noon	Sunny	78.0%	51.8%	46.3%	100.0%	100.0%	0.0%	0 m
		15			73.4%	46.9%	46.0%	84.5%	100.0%	0.0%	0 m
		20			76.1%	54.8%	52.2%	88.0%	100.0%	0.0%	0 m
		25			61.6%	42.1%	43.6%	54.5%	100.0%	0.0%	0 m
		30			70.0%	49.8%	45.8%	59.6%	100.0%	0.0%	0 m
		25	Noon	Cloudy	56.7%	38.6%	36.8%	50.0%	100.0%	0.0%	0 m
				Rainy	66.6%	49.4%	54.2%	65.5%	100.0%	0.0%	0 m
			Sunrise Sunset	Sunny	59.5%	44.5%	54.5%	71.6%	100.0%	0.0%	0 m
					63.8%	48.3%	54.6%	69.0%	100.0%	0.0%	0 m
	Pinhole	10	Noon	Sunny	78.0%	51.8%	46.3%	100.0%	100.0%	0.0%	0 m
		15			73.6%	47.0%	46.4%	85.1%	100.0%	0.0%	0 m
		20			77.7%	56.8%	53.9%	91.4%	100.0%	0.0%	0 m
		25			60.9%	42.4%	44.0%	54.9%	100.0%	0.0%	0 m
		30			74.5%	53.8%	49.1%	64.3%	100.0%	0.0%	0 m
	Fusion	10	Noon	Sunny	78.0%	51.8%	46.3%	100.0%	100.0%	0.0%	0 m
		15			73.4%	46.9%	46.0%	84.5%	100.0%	0.0%	0 m
		20			78.0%	56.1%	53.3%	90.2%	100.0%	0.0%	0 m
		25			66.3%	46.2%	47.0%	59.8%	100.0%	0.0%	0 m
		30			76.2%	55.1%	49.6%	65.9%	100.0%	0.0%	0 m
SIB [420]	Map	10	Noon	Sunny	74.7%	57.4%	46.7%	100.0%	100.0%	0.0%	0 m
		15			45.3%	28.9%	21.2%	47.1%	100.0%	0.0%	0 m
		20			73.7%	59.8%	50.3%	100.0%	100.0%	0.0%	0 m
		25			76.7%	66.2%	58.9%	100.0%	100.0%	0.0%	0 m
		30			83.4%	74.1%	67.8%	100.0%	100.0%	0.0%	0 m
	Fusion	10	Noon	Sunny	74.7%	57.4%	46.7%	100.0%	100.0%	0.0%	0 m
		15			45.3%	28.9%	21.2%	47.1%	100.0%	0.0%	0 m
		20			73.7%	59.8%	50.3%	100.0%	100.0%	0.0%	0 m
		25			76.7%	66.2%	59.0%	100.0%	100.0%	0.0%	0 m
		30			80.6%	70.4%	63.8%	97.5%	100.0%	0.0%	0 m

Table 7.5: Component- and system-level evaluation results of 3 STOP sign attacks. Results are averaged over 10 runs for each configuration.

7.5.4 Simulation Fidelity Evaluation

Although the simulation fidelity drawback is tolerable since (1) the simulation fidelity is evolving and can be improved by our real vehicle setup and (2) practical attacks/defenses should be capable of domain adaptations (§7.5.1), we still hope that today's simulation fidelity is readily usable for AD AI security evaluation purposes. In this section, we thus take the STOP sign attack as an example to concretely understand this.

Experimental setup. We use the SS attack, and calculate the similarity between the STOP sign detection results in the physical world and the simulation environment in our platform. In the physical-world experiments, we print the adversarial STOP sign and overlay it on a portable STOP sign as shown in Fig. 7.4 (c). We have obtained the permit from our institute's transportation department to reserve a dead-end single-lane road for controlled experiments. Correspondingly, we use a simulation environment with similar road geometry. In both settings, the vehicle drives at 10 mph from 300 feet until passes the STOP sign. In the physical-world setting, we use an iPhone 11 mounted on the vehicle windshield to record the driving traces. We collect a total of 20 driving traces with different STOP sign placements in the physical world. Since simulation results are generally quite stable, we only collect one trace in the simulation. We then run object detection on each camera frame to obtain the STOP sign detection confidences in the driving traces. To ensure the traces are synchronized and comparable, we trim and interpolate confidences in the physical-world traces based to a 20 Hz frequency same as in the simulation. We then calculate the Pearson's correlation between the simulation trace and *each* physical-world trace.

Results. Among 20 physical-world driving traces, we find that an average correlation coefficient r of 0.75 (lowest: 0.62, highest: 0.80), and all correlations are statistically significant ($p < 0.05$). For Pearson's correlation, $r > 0.5$ is considered strongly correlated [128]. This shows that at least for such representative AD AI attack type, today's simulation fidelity is

sufficient at least for our AD AI security evaluation purpose.

7.5.5 Educational Usage of *PASS*

Beyond research usage, *PASS* can also be used for educational purposes. We recently organize a Capture The Flag (CTF) event focusing on AD AI security (AutoDriving CTF [5], co-located with DEF CON 29). Specifically, we leverage an earlier version of *PASS* to create simulation-based CTF challenges on GPS/LiDAR spoofing, lane detection attacks, etc. In total, over 100 teams across the globe participated in the CTF. Since *PASS* was provided as a resource for the teams, after the event, we asked the 5 winning teams for feedback on the platform (IRB exempted). Particularly, all 5 teams consider the platform as helpful for solving the challenges. However, due to relatively high resource requirement (e.g., GPUs), 2 teams suggested providing the platform as an online service. This is also the reason why providing remote evaluation services is part of our design goals in §7.5.1.

7.6 Summary

In this work, we perform the first systematization of knowledge of the growing semantic AD AI security research space. We collect and analyze 53 such papers, and systematically taxonomize them based on research aspects critical for the security field. We summarize 6 most substantial scientific gaps based on both quantitative vertically and horizontal comparisons, and provide insights and potential future directions not only at design level, but also at research goal, methodology, and community levels. To address the most critical scientific methodology-level gap, we take the initiative to develop an open-source, uniform, and extensible system-driven evaluation platform *PASS* for the community. We hope that the systematization of knowledge, identified scientific gaps, insights, future direction, and our

community-level evaluation infrastructure building efforts can foster more extensive, realistic, and democratized future research into this critical research space.

Chapter 8

Conclusion and Future Work

This chapter concludes the dissertation by highlighting the contributions in my research and discusses the future directions.

8.1 Conclusion

In this dissertation, I systematically analyze the security of state-of-the-art production-grade AD systems [7, 131] and design 3 highly safety-critical attacks (FusionRipper [335], ROI attack [355], and DRP attack [327]) that can (1) introduce large lateral deviations in MSF-based localization using GPS spoofing alone, which can be further leveraged to (2) fool the traffic light detection in high-level AD systems, and (3) cause unexpected lane departures with carefully crafted benign-looking road dirty patterns for DNN-based ALC in low-level AD systems. On the defense side, I explore practical defense opportunities that are readily-available in CAV systems to defend against existing lateral-direction localization attacks [335] to high-level AD systems and CV data spoofing attacks [119] against intelligent traffic signal control systems. Given the increasing number of research works in AD security, I also sys-

tematize the existing research efforts in recent years by performing the first SoK of semantic AD AI security to help guide future research in this emerging domain. More generally, my research is able to demonstrate that CAV systems' reliance on the physical-layer information inevitably introduces new security challenges that can be leveraged by attackers to cause safety-critical consequences. On the other hand, such CAV systems also contain physical-layer information that can be repurposed for practical defense purposes against existing attacks.

8.2 Future Work

8.2.1 Simultaneous Attacks on Physical-Layer Information

Simultaneous attacks to make MSF attacks deterministic. As shown in the FusionRipper work (§3.3), the take-over vulnerability in MSF is *non-deterministic* because of the real-time dynamic factors such sensor noises and algorithm inaccuracies. In other words, the attacker cannot accurately predict when and where the vulnerable period will occur during the attack. This limits the applicability of the attack, e.g., ambush attacks at attacker-specified locations. To overcome this, a potential direction is simultaneous attacks that aim to increase the sensor noises and algorithm inaccuracies *deterministically* at certain location while launching the GPS spoofing. Specifically, the LiDAR locator performance depends on the road condition and LiDAR point cloud measurement, where both can potentially be influenced by physical-layer attack vectors. For example, the attacker can change the physical world environment by adding/removing road objects; LiDAR spoofing [336, 111] can add new points or remove existing points to/from the point cloud. In such cases, it might be possible to deterministically increase the LiDAR inaccuracy and noise. However, it is still unclear to what extent such attack vectors can influence the point cloud matching in

the LiDAR locator and whether such influence is large enough to enable the take-over effect.

Simultaneous attacks to evade defenses that leverage physical-layer information. For defenses that leverage physical-layer information to cross-validate other cyber- or physical-layer attacks, attackers could launch simultaneous attacks on such physical-layer defense sources to evade the detection. For example, LD^3 and CV defense are vulnerable to simultaneous manipulation of the camera data when launching GPS and data spoofing attacks, respectively. There are multiple possible attack vectors that can manipulate the camera data. In fact, for LD^3, our DRP attack (§4.2) and prior works [283, 210] have demonstrated that perturbations to the road surface are able to control the camera-based lane detection outputs; for CV defense, many physical world adversarial attacks [417, 279, 388] have been demonstrated to be effective against camera-based vehicle detection. However, such simultaneous attacks require addressing the nontrivial challenge of *attack synchronization*. For example, the existing FusionRipper attack to MSF localization is non-deterministic as mentioned above. To evade LD^3, the attack on the camera side needs to not only synchronize the deviation amount but also the deviation timing with the FusionRipper attack. Similarly, to evade CV defense, the attacker needs to adapt the malicious vehicle detection (i.e., vehicle position and velocity) in the camera range with the dynamic out-of-range CVs and RVs such that the spoofing CVs at the far position would be considered as benign CVs.

8.2.2 Security Analyses on Downstream AD AI Components

As shown in Chapter 7, currently there is a substantial lack of works focusing on downstream AI components (e.g., prediction, planning), which are equally (if not more) important compared to upstream ones (e.g., perception, localization) w.r.t. system-level attack effect. Therefore, a future direction is to study the security of those downstream component behaviors leveraging *attacker-controllable physical-layer information*, e.g., road objects

and sensor data. As the downstream components are typically directly related to driving decision-making, the vulnerabilities in them can often directly cause system-level attack consequences, such as overly-aggressive (e.g., those lead to collisions) or overly-conservative (e.g., those lead to commercial service degradation) driving behaviors.

Addressing the system-to-AI semantic gap. Since the downstream AI components are situated deeply in the AD pipeline, it is theoretically more difficult to address the system-to-AI semantic gap (§7.1) than the attacks to the upstream ones. In addition, when applying physical-layer attacks, it is possible that the triggered vulnerability is due to upstream components instead of the downstream ones. Therefore, the security analyses, if done automatically, should ideally be able to accurately attribute the root causes of the vulnerabilities to reduce manual triage efforts. In AD systems, the upstream components are in charge of interpreting the semantics of the driving environment (e.g., ego-vehicle location, obstacle type/dimension/motion, road geometry), and the downstream components make driving decisions based on such semantics. Therefore, a potential solution to address such system-to-AI semantic gap is to directly create/model a driving environment with semantic perturbations, e.g., changing ego-vehicle placement, adding/modifying a vehicle, etc. This can be done in either the physical world or in simulation. However, due to the safety risks and high cost of security testing in the physical world, it is thus more attractive to adopt simulation such as the system-driven evaluation platform *PASS* (§7.5).

8.2.3 Defense Generality Improvements

Generality of LD^3 *to other available road markings.* Although LD^3 can already cover the majority attack scenarios benefit from the non-deterministic nature of state-of-the-art lateral-direction localization attack and is generally available for autonomous driving trucks, the lane line coverage is still a challenge for LD^3, especially in the intersections on urban roads.

To address this, a potential improvement is to include other available road markings, e.g., stop lines [246] and crosswalk markings [94], to localize the vehicle and to detect localization deviations. However, it is still unclear how pervasive such road markings are and more importantly, how to seamlessly switch between the usage of different road markings in LD^3 to maintain consistent defense capability.

Generality of CV defense to vehicle-side CV applications. In Chapter 6, we demonstrate the effectiveness of our CV defense against two representative congestion attacks on the infrastructure-side CV application. However, since the defense is based on a general principle that uses physical-layer information to cross validate the cyber-layer information, it presents as a promising spoofing defense solution for vehicle-side CV applications, e.g., cooperative adaptive cruise control [337], vehicle platooning [98]. However, for such applications, the coverage of infrastructure-side sensors would be a main concern, since the existing infrastructure-side sensors are mostly installed in intersections due to their direct benefit to signal control. Since the minimum required signal spacing for major arterial roads is 1/2 miles [384, 272, 342], the intersection sensors may already able to provide a good coverage to build the physical root-of-trust. Therefore, a potential direction is to systematically explore the tradeoff between the defense effectiveness and infrastructure-side sensor coverage to assist future CV infrastructure design.

8.2.4 System-Driven Evaluation Platform for AD AI Security

In Chapter 7, we also identify that existing semantic AD AI security research works exhibits a general lack of system-level evaluation (§7.4.1), which is especially critical for measuring meaningful scientific progress. To address this critical gap, we took the initiative to develop an open-source, uniform, and extensible system-driven evaluation platform, named *PASS* [334]. However, as a community-level effort, the current PASS design/implementation

is still incomplete. In the future, more research/engineering efforts are needed to extend the security interface design/implementation, integrate more AD AI attacks/defenses, include more diverse driving scenarios/metrics for evaluation. To foster community-level engagement, we plan to continue organizing annual CTF events upon the platform for education purposes and to encourage open-source contributions.

Bibliography

[1] 2019 MKZ. https://www.lincoln.com/services/assets/Brochure?make=Lincoln&model=MKZ&year=2019.

[2] 2020 Accord Hybrid Owner's Manual. http://techinfo.honda.com/rjanisis/pubs/OM/AH/ATWA2020OM/enu/ATWA2020OM.PDF.

[3] 40+ Corporations Working On Autonomous Vehicles. https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list.

[4] Apollo Data Open Platform. http://apollo.auto/index.html.

[5] AutoDriving CTF. https://autodrivingctf.org/.

[6] Autoware Wiki. https://gitlab.com/autowarefoundation/autoware.ai/autoware/-/wikis/Overview.

[7] Baidu Apollo. https://github.com/ApolloAuto/apollo.

[8] Baidu Apollo Perception Module. https://github.com/ApolloAuto/apollo/tree/master/modules/perception.

[9] Baidu Launches Apollo Go Robotaxis In Beijing, Cangzhou, and Changsha. https://cleantechnica.com/2020/09/11/baidu-launches-apollo-go-robotaxis-in-beijing-cangzhou-changsha/.

[10] Baidu rolls out China's first paid, driverless taxi service. https://techwireasia.com/2021/05/baidu-rolls-out-chinas-first-paid-driverless-taxi-service/.

[11] Baidu's self-driving bus, Robobus, is on cusp of commercial operation. https://www.globaltimes.cn/page/202104/1220826.shtml.

[12] California Commercial Driver Handbook: Section 2 – Driving Safely. https://www.dmv.ca.gov/portal/dmv/detail/pubs/cdl_htm/sec2.

[13] California Vehicle Code 21460. https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=VEH&sectionNum=21460.

[14] California Vehicle Code 21663. https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=VEH&sectionNum=21663.

[15] Cohda Wireless OBU. https://www.cohdawireless.com/solutions/hardware/mk5-obu/.

[16] Connected Vehicle Applications for Mobility. https://www.its.dot.gov/pilots/pilots_mobility.htm.

[17] Dirty Road Patch Attack Project Website. https://sites.google.com/view/cav-sec/drp-attack.

[18] Does your car have automated emergency braking? It's a big fail for pedestrians. https://zd.net/2MoUqpd.

[19] Guide to Lane Departure Warning & Lane Keeping Assist. https://www.consumerreports.org/car-safety/lane-departure-warning-lane-keeping-assist-guide/.

[20] HD Maps: New Age Maps Powering Autonomous Vehicles. https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/.

[21] Hyundai plans to launch a free robot taxi service in California. https://www.theverge.com/2019/10/25/20932237/hyundai-self-driving-car-ride-hail-irvine-date.

[22] Introduction to Self-Driving Cars. https://www.coursera.org/learn/intro-self-driving-cars.

[23] Lyft and Aptiv Have Completed 50,000 Self-Driving Car Rides in Las Vegas. https://www.cnet.com/roadshow/news/lyft-aptiv-self-driving-car-50k-rides/.

[24] Nuro can now operate and charge for autonomous delivery services in California. https://tcrn.ch/3mL70PI.

[25] Project Website for the SoK of Semantic AD AI Security. https://sites.google.com/view/cav-sec/pass.

[26] SAVIOR codebase. https://github.com/Cyphysecurity/SAVIOR.

[27] See What Tesla Autopilot Sees At Night In Rain: Video. https://insideevs.com/news/348362/video-what-tesla-autopilot-sees-night-rain/.

[28] Self-Driving Car Engineer Nanodegree. https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013.

[29] Self-Driving Fundamentals: Featuring Apollo. https://www.udacity.com/course/self-driving-car-fundamentals-featuring-apollo--ud0419.

[30] ShapeShifter. https://github.com/shangtse/robust-physical-attack.

[31] State Estimation and Localization for Self-Driving Cars. https://www.coursera.org/learn/state-estimation-localization-self-driving-cars.

[32] Super Cruise - Hands Free Driving — Cadillac Ownership. https://www.cadillac.com/world-of-cadillac/innovation/super-cruise.

[33] Tesla Autopilot. https://www.tesla.com/autopilot.

[34] Tesla Autopilot Support. https://www.tesla.com/support/autopilot.

[35] Tesla releases new, highly anticipated Traffic Light and Stop Sign Control feature. https://electrek.co/2020/04/24/tesla-autopilot-traffic-light-and-stop-sign-control-feature/.

[36] Tesla Sold 2 Million Electric Cars: First Automaker To Reach Milestone. insideevs.com/news/542197/tesla-sold-2000000-electric-cars/.

[37] Tesla Vulnerable to GNSS Spoofing Attacks. https://www.gpsworld.com/tesla-model-s-and-model-3-vulnerable-to-gnss-spoofing-attacks/.

[38] The Autonomous Truck Revolution Is Right Around The Corner. https://www.forbes.com/sites/stevebanker/2021/05/11/the-autonomous-truck-revolution-is-right-around-the-corner/?sh=3d0022222c96.

[39] Tinkla: Tinkering with Tesla. https://tinkla.us/.

[40] Toyota 2020 RAV4 Owner's Manual. https://www.toyota.com/t3Portal/document/om-s/OM0R024U/xhtml/OM0R024U.html.

[41] Uber is Bringing its Self-Driving Cars to Dallas. https://www.theverge.com/2019/9/17/20870969/uber-self-driving-car-testing-dallas.

[42] Uber Self-Driving Car Crash: What Really Happened. https://www.forbes.com/sites/meriameberboucha/2018/05/28/uber-self-driving-car-crash-what-really-happened.

[43] UPS joins race for future of delivery services by investing in self-driving trucks. https://abcnews.go.com/Business/ups-joins-race-future-delivery-services-investing-driving/story?id=65014414.

[44] Video demo for the FusionRipper attack in the paper. https://sites.google.com/view/cav-sec/fusionripper.

[45] Wall Street Journal: Tesla's Driver-Assistance Autopilot Draws Safety Scrutiny. https://www.wsj.com/articles/teslas-driver-assistance-autopilot-draws-safety-scrutiny-11582672837?mod=article_inline.

[46] Walmart First to Deliver Driverless Middle Mile. https://multichannelmerchant.com/operations/walmart-first-to-deliver-driverless-middle-mile/.

[47] Waymo has launched its commercial self-driving service in Phoenix - and it's called 'Waymo One'. https://www.businessinsider.com/waymo-one-driverless-car-service-launches-in-phoenix-arizona-2018-12.

[48] Waymo's Self-Driving Cars Are Near: Meet the Teen Who Rides One Every Day. https://www.bloomberg.com/news/features/2018-07-31/inside-the-life-of-waymo-s-driverless-test-family.

[49] Waymo's driverless taxi service can now be accessed on Google Maps. https://techcrunch.com/2021/06/03/waymos-driverless-taxi-service-can-now-be-accessed-on-google-maps/.

[50] Waymo's next-generation self-driving system can 'see' a stop sign 500 meters away. https://www.theverge.com/2020/3/4/21165014/waymo-fifth-generation-self-driving-radar-camera-lidar-jaguar-ipace.

[51] What Are Connected Vehicles and Why Do We Need Them? https://www.its.dot.gov/cv_basics/cv_basics_what.htm.

[52] What Are the Benefits of Connected Vehicles? https://www.its.dot.gov/cv_basics/cv_basics_benefits.htm.

[53] California Penal Code 594. https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=PEN&sectionNum=594, 1872.

[54] TuSimple Lane Detection Challenge. https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection, 2017.

[55] A Policy on Geometric Design of Highways and Streets, 7th Edition. *AASHTO*, 2018.

[56] Hands-on with Comma.ai's Add-on Level 2 Autonomous Tech. https://www.cnet.com/roadshow/news/hands-on-with-comma-ais-add-on-level-2-autonomous-tech/, 2018.

[57] Waymo Has Launched its Commercial Self-Driving Service in Phoenix — and it's Called 'Waymo One'. https://www.businessinsider.com/waymo-one-driverless-car-service-launches-in-phoenix-arizona-2018-12, 2018.

[58] Adhesive Patch can Seal Potholes and Cracks on the Road. https://www.startupselfie.net/2019/05/07/american-road-patch-seals-potholes-road-cracks/, 2019.

[59] Experimental Security Research of Tesla Autopilot. https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf, 2019.

[60] GM Cadillac CT6 Owner's Manual. https://www.cadillac.com/content/dam/cadillac/na/us/english/index/ownership/technology/supercruise/pdfs/2020-cad-ct6-owners-manual.pdf, 2019.

[61] Nissan Rogue Sports Owner's Manual. https://www.nissan-cdn.net/content/dam/Nissan/pr/Owners-manuals/rogue-sport/2019-RogueSport-owner-manual.pdf, 2019.

[62] Ford Escape Owner's Manual. https://www.ford.com/support/vehicle/escape/2020/owner-manuals/, 2020.

[63] Hyundai Sonata Owner's Manual. https://owners.hyundaiusa.com/us/en/resources/manuals-warranties.html, 2020.

[64] Is a $1000 Aftermarket Add-On as Capable as Tesla's Autopilot and Cadillac's Super Cruise? https://www.caranddriver.com/features/a30341053/self-driving-technology-comparison/, 2020.

[65] KIA Seltos Owner's Manual. https://www.kia.ca/content/ownership/ownersmanual/21seltos.pdf, 2020.

[66] Lane Keeping Assist System Using Model Predictive Control. https://www.mathworks.com/help/mpc/ug/lane-keeping-assist-system-using-model-predictive-control.html, 2020.

[67] Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/, 2020.

[68] Volvo XC90 Owner's Manual. https://volvornt.harte-hanks.com/manuals/2020/XC90_OwnersManual_MY20_en-US_TP29159[1].pdf, 2020.

[69] We Hit the Road with Comma.ai's Assisted-Driving Tech at CES 2020. https://www.cnet.com/roadshow/news/comma-ai-assisted-driving-george-hotz-ces-2020/, 2020.

[70] Car Crashes Get Predictable With Ford 'RoadSafe' Dashboard Technology. https://fordauthority.com/2021/09/car-crashes-get-predictable-with-ford-roadsafe-dashboard-technology/, 2021.

[71] A. Abdo, S. M. B. Malek, Z. Qian, Q. Zhu, M. Barth, and N. Abu-Ghazaleh. Application Level Attacks on Connected Vehicle Protocols. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, pages 459–471, 2019.

[72] H. Abdullah, M. S. Rahman, W. Garcia, K. Warren, A. S. Yadav, T. Shrimpton, and P. Traynor. Hear" No Evil", See" Kenansville"*: Efficient and Transferable Black-Box Attacks on Speech Recognition and Voice Identification systems. In *IEEE S&P*. IEEE, 2021.

[73] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. In *IEEE S&P*, pages 730–747. IEEE, 2021.

[74] W. G. Aguilar, V. S. Salcedo, D. S. Sandoval, and B. Cobeña. Developing of a Video-Based Model for UAV Autonomous Navigation. In *Latin American Workshop on Computational Neuroscience*, pages 94–105. Springer, 2017.

[75] K. Ahn, H. Rakha, and D. K. Hale. Multi-Modal Intelligent Traffic Signal Systems (MMITSS) impacts assessment. Technical report, United States. Department of Transportation, 2015.

[76] S. Ahn, M. J. Cassidy, and J. Laval. Verification of a Simplified Car-Following Theory. *Transportation Research Part B: Methodological*, 38(5):431–440, 2004.

[77] N. Akhtar and A. Mian. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access*, 2018.

[78] D. M. Akos. Who's Afraid of the Spoofer? GPS/GNSS Spoofing Detection via Automatic Gain Control (AGC). *NAVIGATION: Journal of the Institute of Navigation*, 59(4):281–290, 2012.

[79] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Robust and Secure UAV Navigation Using GNSS, Phased-Array Radio System and Inertial Sensor Fusion. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1338–1345. IEEE, 2018.

[80] E. Allak, R. Jung, and S. Weiss. Covariance Pre-Integration for Delayed Measurements in Multi-Sensor Fusion. In *IROS*. IEEE, 2019.

[81] M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt. Security Vulnerabilities of Connected Vehicle Streams and Their Impact on Cooperative Driving. *IEEE Communications Magazine*, 53(6):126–132, 2015.

[82] K. H. Ang, G. Chong, and Y. Li. PID Control System Analysis, Design, and Technology. *CST*, 2005.

[83] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google Street View: Capturing the World at Street Level. *Computer*, 43(6):32–38, 2010.

[84] Baidu Apollo Estimates Object Distances based on Pinhole Camera Model. `https://github.com/ApolloAuto/apollo/blob/v6.0.0/modules/perception/camera/lib/obstacle/transformer/multicue/obj_mapper.cc#L64`.

[85] Aptiv, Audi, Baidu, BMW, Continental, Daimler, Fiat Chrysler Automobiles, HERE, Infineon, Intel and Volkswagen. Safety First for Automated Driving. `https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf`, 2019.

[86] G. D. Arana, M. Joerger, and M. Spenko. Efficient Integrity Monitoring for KF-based Localization. In *ICRA*. IEEE, 2019.

[87] Armen Hareyan. Baidu To Operate 3,000 Driverless Apollo Go Robotaxis in 30 Cities in 3 Years. https://www.torquenews.com/1/baidu-operate-3000-driverless-apollo-go-robotaxies-30-cities-3-years.

[88] S. Arnold and L. Medagoda. Robust Model-Aided Inertial Localization for Autonomous Underwater Vehicles. In *ICRA*. IEEE, 2018.

[89] A. Athalye, N. Carlini, and D. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning (ICML)*, 2018.

[90] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing Robust Adversarial Examples. In *ICML*, 2018.

[91] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver. Autonomous Vehicles: Challenges, Opportunities, and Future Implications for Transportation Policies. *Journal of modern transportation*, 2016.

[92] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun. Deep Multi-Sensor Lane Detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3102–3109, 2018.

[93] Baidu. Apollo Planning Module. https://github.com/ApolloAuto/apollo/tree/master/modules/planning.

[94] O. Bailo, S. Lee, F. Rameau, J. S. Yoon, and I. S. Kweon. Robust Road Marking Detection and Recognition Using Density-Based Grouping and Machine Learning Techniques. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 760–768. IEEE, 2017.

[95] R. Baker and I. Martinovic. Losing the Car Keys: Wireless PHY-Layer Insecurity in EV Charging. In *USENIX Security*, 2019.

[96] M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. *arXiv preprint arXiv:1812.03079*, 2018.

[97] C. Becker, L. J. Yount, S. Rozen-Levy, and J. D. Brewer. Functional Safety Assessment of an Automated Lane Centering System. In *National Highway Traffic Safety Administration*, 2018.

[98] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa. Overview of Platooning Systems. In *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

[99] E. Berger. CSRankings. http://csrankings.org/.

[100] M. Bertoncello and D. Wee. Ten ways autonomous driving could redefine the automotive world. *McKinsey & Company*, 6, 2015.

[101] J. Bhatti and T. E. Humphreys. Hostile Control of Ships via False GPS Signals: Demonstration and Detection. *NAVIGATION: Journal of the Institute of Navigation*, 2017.

[102] P. Biber and W. Straßer. The Normal Distributions Transform: A New Approach to Laser Scan Matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 3, pages 2743–2748. IEEE, 2003.

[103] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang. Attacking Vision-based Perception in End-to-End Autonomous Driving Models. *JSA*, 2020.

[104] A. Boora, I. Ghosh, and S. Chandra. Identification of Free Flowing Vehicles on Two Lane Intercity Highways under Heterogeneous Traffic condition. *Transportation Research Procedia*, 21:130–140, 2017.

[105] M. Brossard and S. Bonnabel. Learning Wheel Odometry and IMU Errors for Localization. In *ICRA*. IEEE, 2019.

[106] M. Brossard, S. Bonnabel, and A. Barrau. Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry. In *IROS*. IEEE, 2018.

[107] M. Brown, J. Crawford, S. Nordstrom, F. Scholl, and F. Mhlanga. Understanding the Presence of Experiential Learning Opportunity Programs in the Information Security Field. In *Proceedings of the 2013 on InfoSecCD'13: Information Security Curriculum Development Conference*, page 53. ACM, 2013.

[108] T. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer. Adversarial Patch. *arXiv:1712.09665*, 2017.

[109] C4ADS. Above Us Only Stars - Exposing GPS Spoofing in Russia and Syria. `https://www.c4reports.org/aboveusonlystars`.

[110] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks. In *IEEE S&P*. IEEE, 2021.

[111] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. In *ACM CCS*, 2019.

[112] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE S&P*, 2017.

[113] N. Carlini and D. Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[114] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial Attacks and Defences: A Survey. *arXiv preprint arXiv:1810.00069*, 2018.

[115] H. F. Chame, M. M. Dos Santos, and S. S. da Costa Botelho. Reliable Fusion of Black-box Estimates of Underwater Localization. In *IROS*, pages 1900–1905. IEEE, 2018.

[116] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security*, volume 4, pages 447–462. San Francisco, 2011.

[117] P.-C. Chen, B.-H. Kung, and J.-C. Chen. Class-Aware Robust Adversarial Training for Object Detection. In *CVPR*, 2021.

[118] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models. In *AISec*, 2017.

[119] Q. A. Chen, Y. Yin, Y. Feng, Z. M. Mao, and H. X. Liu. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *NDSS*, 2018.

[120] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau. ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector. In *ECML PKDD*, pages 52–68. Springer, 2018.

[121] X. Chen, C. Fu, F. Zheng, Y. Zhao, H. Li, P. Luo, and G.-J. Qi. A Unified Multi-Scenario Attacking Network for Visual Object Tracking. In *AAAI*, 2021.

[122] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *CVPR*, 2017.

[123] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun. GeoSim: Realistic Video Simulation via Geometry-Aware Composition for Self-Driving. In *CVPR*, pages 7230–7240, 2021.

[124] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. Kim. Are Self-Driving Cars Secure? Evasion Attacks Against Deep Neural Networks for Steering Angle Prediction. In *IEEE Security and Privacy Workshops (SPW)*, pages 132–137, 2019.

[125] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu. Software-based Realtime Recovery from Sensor Attacks on Robotic Vehicles. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 349–364, 2020.

[126] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *CCS*, pages 801–816, 2018.

[127] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban Environment. In *2013 IEEE International Conference on Robotics and Automation*, pages 1554–1559. IEEE, 2013.

[128] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, 2013.

[129] J. Cohen, E. Rosenfeld, and Z. Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning (ICML)*, pages 1310–1320, 2019.

[130] comma.ai. Announcing the EON Dashcam DevKit. `https://blog.comma.ai/announcing-the-eon-dashcam-devkit/`.

[131] comma.ai. openpilot. `https://github.com/commaai/openpilot`.

[132] K. Cordes, N. Nolte, N. Meine, and H. Broszio. Accuracy evaluation of camera-based vehicle localization. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–7. IEEE, 2019.

[133] What it really costs to turn a car into a self-driving vehicle. `https://qz.com/924212/`.

[134] N. S. Council. *Reference Material for DDC Instructors, 5th Edition*. 2005.

[135] C. Croarkin, P. Tobias, J. Filliben, B. Hembree, W. Guthrie, et al. NIST/SEMATECH e-Handbook of Statistical Methods. *NIST/SEMATECH, July. Available online: http://www. itl. nist. gov/div898/handbook*, 2006.

[136] GM Safety Report 2018. `https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf`.

[137] Cubic. GRIDSMART — Single Camera Solution for Traffic Management. `https://gridsmart.com/products/gridsmart-solution/`.

[138] Curtis Vickers. New roadside LiDAR sensors help build a safer transportation infrastructure. `https://www.unr.edu/nevada-today/news/2020/hao-xu-roadside-lidar`, 2020.

[139] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart. Controlling UAVs with Sensor Input Spoofing Attacks. In *WOOT*, 2016.

[140] F. de Ponte Müller. Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles. *Sensors*, 17(2):271, 2017.

[141] E. Deligne. ARDrone Corruption. *Journal in Computer Virology*, 8(1):15–27, 2012.

[142] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, and Q.-L. Han. Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses. *IEEE Transactions on Industrial Informatics*, 2021.

[143] L. Ding, Y. Wang, K. Yuan, M. Jiang, P. Wang, H. Huang, and Z. J. Wang. Towards Universal Physical Attacks on Single Object Tracking. In *AAAI*, 2021.

[144] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting Adversarial Attacks With Momentum. In *CVPR*, pages 9185–9193, 2018.

[145] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Pearson, 2011.

[146] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[147] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Annual Conference on Robot Learning*, 2017.

[148] Douglas Gettman. DSRC and C-V2X: Similarities, Differences, and the Future of Connected Vehicles. `https://www.kimley-horn.com/dsrc-cv2x-comparison-future-connected-vehicles/`, 2020.

[149] T. Dreossi, A. Donzé, and S. A. Seshia. Compositional Falsification of Cyber-Physical Systems with Machine Learning Components. *Journal of Automated Reasoning*, 63(4):1031–1053, 2019.

[150] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia. VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-based Systems. In *CAV*, 2019.

[151] J. Ebrahimi, D. Lowd, and D. Dou. On Adversarial Examples for Character-Level Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*. Association for Computational Linguistics, 2018.

[152] K. Eckenhoff, P. Geneva, J. Bloecker, and G. Huang. Multi-Camera Visual-Inertial Navigation with Online Intrinsic and Extrinsic Calibration. In *ICRA*, pages 3158–3164. IEEE, 2019.

[153] ETAuto. Europe to add 69 million connected cars during 2020-25: Report. `https://auto.economictimes.indiatimes.com/news/auto-technology/europe-to-add-69-million-connected-cars-during-2020-25-report/79350675`, 2020.

[154] ETH Zürich. Ethzasl MSF Framework. `https://github.com/ethz-asl/ethzasl_msf`.

[155] European GNSS Agency. Report On Road User Needs And Requirements. Technical report, European GNSS Agency, 2019.

[156] A. Evlampev, I. Shapovalov, and S. Gafurov. Map relative localization based on road lane matching with Iterative Closest Point algorithm. In *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, pages 232–236, 2020.

[157] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song. Physical Adversarial Examples for Object Detectors. In *WOOT*, 2018.

[158] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust Physical-World Attacks on Deep Learning Visual Classification. In *CVPR*, 2018.

[159] Federal Highway Administration. Roadway Departure Safety. `https://safety.fhwa.dot.gov/roadway_dept/`.

[160] M. Fellendorf and P. Vortisch. Microscopic traffic flow simulator VISSIM. In *Fundamentals of traffic simulation*, pages 63–93. Springer, 2010.

[161] Z. Feng, N. Guan, M. Lv, W. Liu, Q. Deng, X. Liu, and W. Yi. Efficient Drone Hijacking Detection using Onboard Motion Sensors. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1414–1419. IEEE, 2017.

[162] Z. Feng, N. Guan, M. Lv, W. Liu, Q. Deng, X. Liu, and W. Yi. An Efficient UAV Hijacking Detection Method Using Onboard Inertial Measurement Unit. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(6):1–19, 2018.

[163] U. D. for Transport. *The Official Highway Code Book*. 2015.

[164] L. Franceschi-Bicchierai. Drone Hijacking? That's Just the Start of GPS Troubles. *Retrieved April*, 27:2013, 2012.

[165] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell. *Feedback Control of Dynamic Systems*. 2002.

[166] B. Friedland. *Control System Design: An Introduction to State-Space Methods*. Courier Corporation, 2012.

[167] D. Frossard and R. Urtasun. End-to-end Learning of Multi-sensor 3D Tracking by Detection. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 635–642. IEEE, 2018.

[168] C. Gackstatter, P. Heinemann, S. Thomas, and G. Klinker. Stable Road Lane Model Based on Clothoids. In *Advanced Microsystems for Automotive Applications*, pages 133–143. Springer, 2010.

[169] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In *CVPR*, 2020.

[170] Y. Gao, S. Liu, M. Atia, and A. Noureldin. INS/GPS/LiDAR Integrated Navigation System for Urban and Indoor Environments Using Hybrid Scan Matching Algorithm. *Sensors*, 15(9):23286–23302, 2015.

[171] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès. Lock It and Still Lose It —on the (In)Security of Automotive Remote Keyless Entry Systems. In *USENIX Security*, 2016.

[172] P. Geneva, K. Eckenhoff, and G. Huang. Asynchronous Multi-Sensor Fusion for 3D Mapping and Localization. In *ICRA*. IEEE, 2018.

[173] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[174] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu. Security and Privacy in Cyber-Physical Systems: A Survey of Surveys. *IEEE Design & Test*, 34(4):7–17, 2017.

[175] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.

[176] N. Gosala, A. Bühler, M. Prajapat, C. Ehmke, M. Gupta, R. Sivanesan, A. Gawel, M. Pfeiffer, M. Bürki, I. Sa, et al. Redundant Perception and State Estimation for Reliable Autonomous Racing. In *ICRA*, pages 6561–6567. IEEE, 2019.

[177] P. D. Groves. Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, [Book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27, 2015.

[178] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten. Countering Adversarial Images using Input Transformations. In *International Conference on Learning Representation (ICLR)*, 2018.

[179] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun. DLFuzz: Differential Fuzzing Testing of Deep Learning Systems. In *ESEC/FSE*, 2018.

[180] P. Guo, H. Kim, L. Guan, M. Zhu, and P. Liu. VCIDS: Collaborative intrusion detection of sensor and actuator attacks on connected vehicles. In *International Conference on Security and Privacy in Communication Systems*, pages 377–396. Springer, 2017.

[181] A. Hamdi, M. Müller, and B. Ghanem. SADA: Semantic Adversarial Diagnostic Attacks for Autonomous Applications. In *AAAI*, 2020.

[182] E. Hamilton. JPEG File Interchange Format. 2004.

[183] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.

[184] Z. Hau, K. T. Co, S. Demetriou, and E. C. Lupu. Object Removal Attacks on LiDAR-based 3D Object Detectors. 2021.

[185] L. Head. The Multi Modal Intelligent Traffic Signal System (MMITSS): A Connected Vehicle Dynamic Mobility Application. In *Mid Year Meeting, Traffic Signal Systems Committee, Transportation Research Board, MMITSS. I-95.06 (20). https://tetcoalition. org/wp-content/uploads/2016/03/Head. MMITSS. I-95.06*, volume 20, 2016.

[186] A. B. Hillel, R. Lerner, D. Levi, and G. Raz. Recent Progress in Road and Lane Detection: A Survey. *Machine vision and applications*, 25(3):727–745, 2014.

[187] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In *2007 American Control Conference*, pages 2296–2301. IEEE, 2007.

[188] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS–Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and More*. Springer Science & Business Media, 2007.

[189] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015.

[190] Honda Technology. Honda Demonstrates New "Smart Intersection" Technology. `https://csr.honda.com/2018/10/04/honda-demonstrates-new-smart-intersection-technology/`, 2018.

[191] D. K. Hong, J. Kloosterman, Y. Jin, Y. Cao, Q. A. Chen, S. Mahlke, and Z. M. Mao. AVGuardian: Detecting and Mitigating Publish-Subscribe Overprivilege for Autonomous Vehicle Systems. In *EuroS&P*, 2020.

[192] S. Hu, Q. A. Chen, J. Joung, C. Carlak, Y. Feng, Z. M. Mao, and H. X. Liu. CVShield: Guarding Sensor Data in Connected Vehicle with Trusted Execution Environment. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, pages 1–4, 2020.

[193] S. Hu, Q. A. Chen, J. Sun, Y. Feng, Z. M. Mao, and H. X. Liu. Automated Discovery of Denial-of-Service Vulnerabilities in Connected Vehicle Protocols. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[194] S. Hu, Y. Zhang, S. Laha, A. Sharma, and H. Foroosh. CCA: Exploring the Possibility of Contextual Camouflage Attack on Object Detection. In *ICPR*. IEEE, 2021.

[195] W. Hu, A. T. McCartt, and E. R. Teoh. Effects of red light camera enforcement on fatal crashes in large US cities. *Journal of safety research*, 42(4):277–282, 2011.

[196] L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu. Universal Physical Camouflage Attacks on Object Detectors. In *CVPR*, 2020.

[197] S. E. Huang, Q. A. Chen, Y. Feng, Z. M. Mao, W. Wong, and H. X. Liu. Impact Evaluation of Falsified Data Attacks on Connected Vehicle Based Traffic Signal Control Systems. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, volume 2021, page 25, 2021.

[198] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. Kintner. Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *ION GNSS'08*, 2008.

[199] S. Hussain, P. Neekhara, S. Dubnov, J. McAuley, and F. Koushanfar. WaveGuard: Understanding and Mitigating Audio Adversarial Examples. In *USENIX Security*, 2021.

[200] Ioanna Lykiardopoulou. Britain's first self-driving shuttle bus hits the streets, but scares passengers away. `https://thenextweb.com/news/uk-first-self-driving-shuttle-bus-scares-passengers-away`.

[201] A. Jafarnia-Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle. GPS Vulnerability to Spoofing Threats and a Review of Antispoofing Techniques. *International Journal of Navigation and Observation*, 2012, 2012.

[202] K. Jansen, M. Schäfer, D. Moser, V. Lenders, C. Pöpper, and J. Schmitt. Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 1018–1031. IEEE, 2018.

[203] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim. Complex Urban Dataset with Multi-Level Sensors from Highly Diverse Urban Environments. *The International Journal of Robotics Research*, 38(6):642–657, 2019.

[204] Jerry Hirsch. Aurora Expands Autonomous Trucking Tests in Texas. `https://www.ttnews.com/articles/aurora-expands-autonomous-trucking-tests-texas`.

[205] S. Jha, S. Cui, S. Banerjee, J. Cyriac, T. Tsai, Z. Kalbarczyk, and R. K. Iyer. ML-driven Malware that Targets AV Safety. In *DSN*. IEEE, 2020.

[206] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu. Poltergeist: Acoustic Adversarial Machine Learning against Cameras and Computer Vision. In *IEEE S&P*, 2021.

[207] S. Jia, C. Ma, Y. Song, and X. Yang. Robust Tracking against Adversarial Attacks. In *ECCV*, 2020.

[208] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei. Fooling Detection Alone is Not Enough: Adversarial Attack Against Multiple Object Tracking. In *International Conference on Learning Representations (ICLR)*, 2019.

[209] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei. Fooling Detection Alone is Not Enough: Adversarial Attack against Multiple Object Tracking. In *ICLR*, 2020.

[210] P. Jing, Q. Tang, Y. Du, L. Xue, X. Luo, T. Wang, S. Nie, and S. Wu. Too Good to Be Safe: Tricking Lane Detection in Autonomous Driving with Crafted Perturbations. In *USENIX Security*, 2021.

[211] Jon Barad. Lidar Technology Making Smart Cities a Reality. `https://velodynelidar.com/blog/lidar-technology-making-smart-cities-a-reality/`, 2019.

[212] J. M. Kang, T. S. Yoon, E. Kim, and J. B. Park. Lane-Level Map-Matching Method for Vehicle Localization Using GPS and Camera on a High-Definition Map. *Sensors*, 20(8):2166, 2020.

[213] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware On Board: Enabling Autonomous Vehicles with Embedded Systems. In *ICCPS'18*, pages 287–296. IEEE Press, 2018.

[214] J. Kelly and G. S. Sukhatme. Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-Calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[215] J. B. Kenney. Dedicated Short-Range Communications (DSRC) Standards in the United States. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[216] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys. Unmanned Aircraft Capture and Control via GPS Spoofing. *Journal of Field Robotics*, 2014.

[217] S. Khanafseh, N. Roshan, S. Langel, F.-C. Chan, M. Joerger, and B. Pervan. GPS Spoofing Detection using RAIM with INS Coupling. In *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*, pages 1232–1239. IEEE, 2014.

[218] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Computers & Security*, page 102150, 2021.

[219] Kim Lyons. Chinese startup Pony.ai gets approval to test driverless vehicles in California. `https://www.theverge.com/2021/5/22/22449084/chinese-startup-pony-ai-autonomous-vehicles-california`.

[220] Kim Lyons. Cruise gets permit from California to provide passenger test rides in driverless vehicles. `https://www.theverge.com/2021/6/5/22520227/cruise-permit-california-driverless-autonomous-vehicles`.

[221] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representation (ICLR)*, 2015.

[222] Kirsten Korosec. Waymo's driverless taxi service can now be accessed on Google Maps. `https://techcrunch.com/2021/06/03/waymos-driverless-taxi-service-can-now-be-accessed-on-google-maps/`.

[223] Y. Ko, J. Jun, D. Ko, and M. Jeon. Key Points Estimation and Point Instance Segmentation Approach for Lane Detection. *arXiv:2002.06604*, 2020.

[224] S. Köhler, G. Lovisotto, S. Birnbach, R. Baker, and I. Martinovic. They See Me Rollin': Inherent Vulnerability of the Rolling Shutter in CMOS Image Sensors. In *ACSAC*, 2021.

[225] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[226] Z. Kong, J. Guo, A. Li, and C. Liu. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. In *CVPR*, 2020.

[227] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Examples in the Physical World. *arXiv:1607.02533*, 2016.

[228] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Machine Learning at Scale. *arXiv preprint arXiv:1611.01236*, 2016.

[229] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis. A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications. *IOT-J*, 5(2):829–846, 2018.

[230] A. Laszka, B. Potteiger, Y. Vorobeychik, S. Amin, and X. Koutsoukos. Vulnerability of Transportation Networks to Traffic-Signal Tampering. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[231] C. Laurendeau and M. Barbeau. Threats to Security in DSRC/WAVE. In *International Conference on Ad-Hoc Networks and Wireless*, pages 266–279. Springer, 2006.

[232] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE S&P*, 2019.

[233] B.-H. Lee, J.-H. Song, J.-H. Im, S.-H. Im, M.-B. Heo, and G.-I. Jee. GPS/DR Error Estimation for Autonomous Vehicle Localization. *Sensors*, 15(8):20779–20798, 2015.

[234] J.-W. Lee and B. Litkouhi. A Unified Framework of the Automated Lane Centering/Changing Control for Motion Smoothness Adaptation. In *International IEEE Conference on Intelligent Transportation Systems*, pages 282–287, 2012.

[235] S. Lee, Y. Cho, and B.-C. Min. Attack-Aware Multi-Sensor Integration Algorithm for Autonomous Vehicle Navigation Systems. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3739–3744. IEEE, 2017.

[236] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Robotics: science and systems*, volume 4, page 1. Citeseer, 2007.

[237] J. Levinson and S. Thrun. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4372–4378. IEEE, 2010.

[238] LG. LGSVL Simulator: An Autonomous Vehicle Simulator. `https://github.com/lgsvl/simulator`.

[239] J. Li, X. Mei, D. Prokhorov, and D. Tao. Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):690–703, 2016.

[240] J. Li, F. Schmidt, and Z. Kolter. Adversarial Camera Stickers: A Physical Camera-Based Attack on Deep Learning Systems. In *International Conference on Machine Learning*, pages 3896–3904, 2019.

[241] S. Li, A. Neupane, S. Paul, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and A. Swami. Stealthy Adversarial Perturbations Against Real-Time Video Classification Systems. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[242] S. Li, S. Zhu, S. Paul, A. Roy-Chowdhury, C. Song, S. Krishnamurthy, A. Swami, and K. S. Chan. Connecting the Dots: Detecting Adversarial Perturbations Using Context Inconsistency. In *ECCV*, 2020.

[243] Y. Li, C. Wen, F. Juefei-Xu, and C. Feng. Fooling LiDAR Perception via Adversarial Trajectory Perturbation. 2021.

[244] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, pages 641–656, 2018.

[245] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[246] G.-T. Lin, P. S. Santoso, C.-T. Lin, C.-C. Tsai, and J.-I. Guo. Stop Line Detection and Distance Measurement for Road Intersection based on Deep Learning Neural Network. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 692–695. IEEE, 2017.

[247] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang. Deepsec: A Uniform Platform for Security Analysis of Deep Learning Model. In *IEEE Symposium on Security and Privacy (SP)*, pages 673–690, 2019.

[248] J. Liu and J. Park. "Seeing is not Always Believing": Detecting Perception Error Attacks Against Autonomous Vehicles. *TDSC*, 2021.

[249] W. Liu, C. Kwon, I. Aljanabi, and I. Hwang. Cyber Security Analysis for State Estimators in Air Traffic Control Systems. In *AIAA Guidance, Navigation, and Control Conference*, page 4929, 2012.

[250] X. Liu, B. Luo, A. Abdo, N. Abu-Ghazaleh, and Q. Zhu. Securing Connected Vehicle Applications with an Efficient Dual Cyber-Physical Blockchain Framework. *arXiv preprint arXiv:2102.07690*, 2021.

[251] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning Attack on Neural Networks. In *NDSS*, 2018.

[252] H. Loeb, A. Belwadi, J. Maheshwari, and S. Shaikh. Age and Gender Differences in Emergency Takeover from Automated to Manual Driving on Simulator. *Traffic injury prevention*, pages 1–3, 2019.

[253] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic. SLAP: Improving Physical Adversarial Examples with Short-Lived Adversarial Perturbations. In *USENIX Security*, 2021.

[254] D. Lu, V. C. Jammula, S. Como, J. Wishart, Y. Chen, and Y. Yang. CAROM–Vehicle Localization and Traffic Scene Reconstruction from Monocular Cameras on Road Infrastructures. In *2021 International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[255] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. In *CVPR Workshop of Negative Results in Computer Vision*, 2017.

[256] A. Luo. Drones Hijacking. *DEF CON, Paris, France*, 2016.

[257] M. Luo, A. C. Myers, and G. E. Suh. Stealthy Tracking of Autonomous Vehicles with Cache Side Channels. In *USENIX Security*, 2020.

[258] Q. Luo, Y. Cao, J. Liu, and A. Benslimane. Localization and Navigation in Autonomous Driving: Threats and Countermeasures. *IEEE Wireless Communications*, 26(4):38–45, 2019.

[259] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim. Multiple Object Tracking: A Literature Review. *AI*, 2020.

[260] Lyft. Semantic Maps for Autonomous Vehicles. `https://medium.com/lyftself-driving/semantic-maps-for-autonomous-vehicles-470830ee28b6`.

[261] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart. A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation. In *IROS*, 2013.

[262] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representation (ICLR)*, 2018.

[263] J. Magiera and R. Katulski. Detection and Mitigation of GPS Spoofing Based on Antenna Array Processing. *Journal of applied research and technology*, 13(1):45–57, 2015.

[264] Y. Man, M. Li, and R. Gerdes. GhostImage: Remote Perception Attacks against Camera-based Image Classification Systems. In *RAID*, 2020.

[265] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun. LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World. In *CVPR*, 2020.

[266] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli. GOMSF: Graph-Optimization based Multi-Sensor Fusion for Robust UAV Pose Estimation. In *ICRA*, pages 1421–1428. IEEE, 2018.

[267] MathWorks. System Identification Toolbox. `https://www.mathworks.com/products/sysid.html`.

[268] A. D. May. *Traffic Flow Fundamentals*. Transportation Research Board, 1990.

[269] N. Medeiros, N. Ivaki, P. Costa, and M. Vieira. Software Metrics as Indicators of Security Vulnerabilities. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 216–227. IEEE, 2017.

[270] H. Merten. The Three-Dimensional Normal-Distributions Transform. *threshold*, 10:3, 2008.

[271] M. Miiller, F. Steidle, M. J. Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomic, and W. Stürzl. Robust Visual-Inertial State Estimation with Multiple Odometries and Efficient Mapping on an MAV with Ultra-Wide FOV Stereo Vision. In *IROS*, pages 3701–3708. IEEE, 2018.

[272] Missouri Department of Transportation. 940.6 Traffic Signal Spacing. `https://epg.modot.org/index.php/940.6_Traffic_Signal_Spacing`.

[273] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. False Data Injection Attacks Against State Estimation in Wireless Sensor Networks. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5967–5972. IEEE, 2010.

[274] Y. Mo and B. Sinopoli. False Data Injection Attacks in Control Systems. In *Preprints of the 1st workshop on Secure Control Systems*, pages 1–6, 2010.

[275] S. Moridpour, M. Sarvi, and G. Rose. Lane Changing Models: A Critical Review. *Transportation letters*, 2(3):157–173, 2010.

[276] T. Multerer, A. Ganis, U. Prechtel, E. Miralles, A. Meusling, J. Mietzner, M. Vossiek, M. Loghi, and V. Ziegler. Low-Cost Jamming System Against Small Drones Using a 3D MIMO Radar based Tracking. In *EURAD*. IEEE, 2017.

[277] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[278] K. Nakashima and R. Kurazume. Learning to Drop Points for LiDAR Scan Synthesis. In *IROS*. IEEE, 2021.

[279] K. K. Nakka and M. Salzmann. Indirect Local Attacks for Context-Aware Semantic Segmentation Networks. In *ECCV*, 2020.

[280] S. Narain, A. Ranganathan, and G. Noubir. Security of GPS/INS based On-Road Location Tracking Systems. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

[281] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici. Drones' Cryptanalysis-Smashing Cryptography with a Flicker. In *IEEE S&P*, 2019.

[282] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici. SoK: Security and Privacy in the Age of Commercial Drones. In *IEEE S&P*, pages 73–90. IEEE, 2021.

[283] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici. Phantom of the ADAS: Securing Advanced Driver-Assistance Systems from Split-Second Phantom Attacks. In *ACM CCS*, 2020.

[284] National Association of City Transportation Officials (NACTO). Vehicle Stopping Distance and Time. `https://nacto.org/docs/usdg/vehicle_stopping_distance_and_time_upenn.pdf`.

[285] F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo. Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks. In *WACV*, 2022.

[286] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards End-to-End Lane Detection: an Instance Segmentation Approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[287] G. F. Newell. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205, 2002.

[288] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley. GPS Software Attacks. In *Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS)*, 2012.

[289] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim. Tractor Beam: Safe-hijacking of Consumer Drones with Adaptive GPS Spoofing. *ACM Transactions on Privacy and Security (TOPS)*, 22(2):1–26, 2019.

[290] NovAtel. NovAtel SPAN on ProPak6 Datasheet. `https://www.novatel.com`.

[291] NYC Connected Vehicle Project. Connected Vehicle technology is coming to the streets of New York City! This technology holds the potential to make our streets safer and smarter. `https://www.cvp.nyc/`.

[292] A. A. of State Highway and T. O. (AASHTO). *Policy on Geometric Design of Highways and Streets (7th Edition)*. American Association of State Highway and Transportation Officials (AASHTO), 2018.

[293] M. O'Kelly, H. Abbas, and R. Mangharam. Computer-Aided Design for Safe Autonomous Vehicles. In *RWS*, 2017.

[294] Comma AI openpilot. `https://github.com/commaai/openpilot`.

[295] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[296] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo. A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Transactions on Emerging Topics in Computing*, 2016.

[297] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[298] P. Papadimitratos and A. Jovanovic. GNSS-based Positioning: Attacks and Countermeasures. In *MILCOM 2008-2008 IEEE Military Communications Conference*, pages 1–7. IEEE, 2008.

[299] A. Papathanassiou and A. Khoryaev. Cellular V2X as the Essential Enabler of Superior Global Connected Transportation Services. *IEEE 5G Tech Focus*, 1(2):1–2, 2017.

[300] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and Privacy in Machine Learning. In *EuroS&P*. IEEE, 2018.

[301] K. Pei, Y. Cao, J. Yang, and S. Jana. Deepxplore: Automated Whitebox Testing of Deep Learning Systems. In *SOSP*, pages 1–18, 2017.

[302] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and Lidar. *Black Hat Europe*, 11:2015, 2015.

[303] R. Piché. Online Tests of Kalman Filter Consistency. *International Journal of Adaptive Control and Signal Processing*, 30(1):115–124, 2016.

[304] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *IEEE S&P*, pages 1332–1349. IEEE, 2020.

[305] S. Piperakis, D. Kanoulas, N. G. Tsagarakis, and P. Trahanias. Outlier-Robust State Estimation for Humanoid Robots. In *IROS*. IEEE, 2019.

[306] F. Poggenhans, N. O. Salscheider, and C. Stiller. Precise Localization in High-definition Road Maps for Urban Regions. In *IROS*, pages 2167–2174. IEEE, 2018.

[307] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle. The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles? In *IEEE intelligent vehicles symposium (IV)*. IEEE, 2017.

[308] Police Radar Information Center. Vehicle Acceleration and Braking Parameters. `https://copradar.com/chapts/references/acceleration.html`.

[309] Praharsha Anand. UC Irvine selects Velodyne Lidar's traffic-monitoring solution. `https://www.itpro.com/technology/smart-city/361047/uc-irvine-selects-velodyne-lidars-traffic-monitoring-solution`, 2021.

[310] M. L. Psiaki and T. E. Humphreys. GNSS Spoofing and Detection. *Proceedings of the IEEE*, 104(6):1258–1270, 2016.

[311] PTV Group. PTV Vissim - Traffic Simulation Software. `https://www.ptvgroup.com/en/solutions/products/ptv-vissim/`.

[312] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha. Securing Connected & Autonomous Vehicles: Challenges Posed by Adversarial Machine Learning and the Way Forward. *IEEE COMST*, 2020.

[313] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *USENIX Security*, 2020.

[314] A. Raghunathan, J. Steinhardt, and P. Liang. Certified Defenses Against Adversarial Examples. 2018.

[315] R. Rajamani. *Vehicle Dynamics and Control*. Springer Science & Business Media, 2011.

[316] A. Ranganathan, H. Ólafsdóttir, and S. Capkun. SPREE: A Spoofing Resistant GPS Receiver. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016.

[317] Raspberry Pi Foundation. Raspberry Pi 3 Model B. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[318] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *CVPR*, pages 7263–7271, 2017.

[319] T. G. Reid, S. E. Houts, R. Cammarata, G. Mills, S. Agarwal, A. Vora, and G. Pandey. Localization Requirements for Autonomous Vehicles. *arXiv preprint arXiv:1906.01061*, 2019.

[320] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin. The Security of Autonomous Driving: Threats, Defenses, and Future Directions. *IEEE*, 2019.

[321] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[322] Richalet, J. and Rault, A. and Testud, J. L. and Papon, J. Model Predictive Heuristic Control. *Automatica*, 14:413–428, 1978.

[323] N. Rodday. Hacking a Professional Drone. *Black Hat Asia*, 2016.

[324] R. W. Rothery. Car Following Models. *Trac Flow Theory*, 1992.

[325] Ryan Wu. C-V2X automotive tech brings enhanced safety and efficiency to China's roads. `https://www.qualcomm.com/news/onq/2021/03/02/c-v2x-brings-enhanced-safety-and-efficiency-chinas-roads`, 2021.

[326] SAE On-Road Automated Vehicle Standards Committee and others. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. *SAE International: Warrendale, PA, USA*, 2021.

[327] T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. In *USENIX Security*, 2021.

[328] A. Sayles, A. Hooda, M. Gupta, R. Chatterjee, and E. Fernandes. Invisible Perturbations: Physical Adversarial Examples Exploiting the Rolling Shutter Effect. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14666–14675, 2021.

[329] H. Schafer, E. Santana, A. Haden, and R. Biasini. A Commute in Data: The comma2k19 Dataset. *arXiv:1812.05752*, 2018.

[330] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[331] M. Schreiber, H. Königshof, A.-M. Hellmund, and C. Stiller. Vehicle Localization with Tightly Coupled GNSS and Visual Odometry. In *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016.

[332] S. A. Seshia, S. Jha, and T. Dreossi. Semantic Adversarial Deep Learning. *IEEE Design & Test*, 37(2):8–18, 2020.

[333] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on state-of-the-art Face Recognition. In *ACM CCS*, 2016.

[334] J. Shen, N. Wang, Z. Wan, Y. Luo, T. Sato, Z. Hu, X. Zhang, S. Guo, Z. Zhong, K. Li, et al. SoK: On the Semantic AI Security in Autonomous Driving. *arXiv preprint arXiv:2203.05314*, 2022.

[335] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing. In *USENIX Security*, 2020.

[336] H. Shin, D. Kim, Y. Kwon, and Y. Kim. Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications. In *CHES*, pages 445–467. Springer, 2017.

[337] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis. Cooperative adaptive cruise control: Definitions and operating concepts. *Transportation Research Record*, 2489(1):145–152, 2015.

[338] P. Smuda, R. Schweiger, H. Neumann, and W. Ritter. Multiple Cue Data Fusion With Particle Filters for Road Course Detection in Vision Systems. In *IEEE Intelligent Vehicles Symposium (IV)*, 2006.

[339] J. Solà. Quaternion Kinematics for the Error-State Kalman Filter. *arXiv preprint arXiv:1711.02508*, 2017.

[340] A. Soloviev. Tight Coupling of GPS, Laser Scanner, and Inertial Measurements for Navigation in Urban Environments. In *IEEE/ION Position, Location and Navigation Symposium*. IEEE, 2008.

[341] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim. Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors. In *USENIX Security*, pages 881–896, 2015.

[342] South Carolina Department of Transportation. Access and Roadside Management Standards (ARMS Manual). `https://www.scdot.org/business/pdf/permits-ARMS_2008.pdf`.

[343] W. Stallings, L. Brown, M. D. Bauer, and M. Howard. *Computer Security: Principles and Practice*, volume 2. Pearson Upper Saddle River, 2012.

[344] W. J. Stein and T. R. Neuman. Mitigation Strategies for Design Exceptions. Technical report, United States. Federal Highway Administration. Office of Safety, 2007.

[345] J. Su, J. He, P. Cheng, and J. Chen. A Stealthy GPS Spoofing Strategy for Manipulating the Trajectory of an Unmanned Aerial Vehicle. *IFAC-PapersOnLine*, 49(22):291–296, 2016.

[346] J. K. Suhr, J. Jang, D. Min, and H. G. Jung. Sensor Fusion-based Low-Cost Vehicle Localization System for Complex Urban Environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1078–1086, 2016.

[347] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao. Towards Robust LiDAR-based Perception in Autonomous Driving: General Black-box Adversarial Sensor Attack and Countermeasures. In *USENIX Security*, 2020.

[348] M. Sun, Y. Man, M. Li, and R. Gerdes. SVM: Secure Vehicle Motion Verification with a Single Wireless Receiver. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 65–76, 2020.

[349] Z. Sun, S. Balakrishnan, L. Su, A. Bhuyan, P. Wang, and C. Qiao. Who Is in Control? Practical Physical Layer Attack and Defense for mmWave-Based Sensing in Autonomous Vehicles. *IEEE Transactions on Information Forensics and Security*, 2021.

[350] Cadillac Super Cruise. `https://www.cadillac.com/world-of-cadillac/innovation/super-cruise`.

[351] SVL Digital Twin. `https://www.svlsimulator.com/docs/digital-twin/gomentum-dtl/`.

[352] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014.

[353] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad. A Survey of End-to-End Driving: Architectures and Training Methods. *TNNLS*, 2020.

[354] S. Tanaka, K. Yamada, T. Ito, and T. Ohkawa. Vehicle Detection Based on Perspective Transformation Using Rear-View Camera. *Hindawi Publishing Corporation International Journal of Vehicular Technology*, 9, 03 2011.

[355] K. Tang, J. S. Shen, and Q. A. Chen. Fooling Perception via Location: A Case of Region-of-Interest Attacks on Traffic Light Detection in Autonomous Driving. In *NDSS Workshop AutoSec*, 2021.

[356] Ç. Tanıl, S. Khanafseh, and B. Pervan. Detecting Global Navigation Satellite System Spoofing Using Inertial Sensing of Aircraft Disturbance. *Journal of Guidance, Control, and Dynamics*, 40(8):2006–2016, 2017.

[357] Z. Tao, P. Bonnifait, V. Fremont, and J. Ibanez-Guzman. Mapping and Localization Using GPS, Lane Markings and Proprioceptive Sensors. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 406–412. IEEE, 2013.

[358] Teledyne FLIR. Teledyne FLIR TrafiSense AI: AI-Powered Thermal Traffic Sensor. `https://www.flir.com/products/trafisense-ai/`.

[359] Tesla. Autopilot. `https://www.tesla.com/autopilot`.

[360] Tesla Autopilot Traffic Light and Stop Sign Control. `https://www.tesla.com/ownersmanual/modely/en_eu/GUID-A701F7DC-875C-4491-BC84-605A77EA152C.html`.

[361] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT press, 2005.

[362] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. In *ICSE*, 2018.

[363] J.-A. Ting, E. Theodorou, and S. Schaal. A Kalman Filter for Robust Outlier Detection. In *IROS*. IEEE, 2007.

[364] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun. On the Requirements for Successful GPS Spoofing Attacks. In *ACM conference on Computer and Communications Security (CCS)*, 2011.

[365] TrafficVision. Traffic Intelligence from Video. `http://www.trafficvision.com/`.

[366] F. Tramer, N. Carlini, W. Brendel, and A. Madry. On Adaptive Attacks to Adversarial Example Defenses. *arXiv:2002.08347*, 2020.

[367] Transoft. TrafxSAFE Connect - Real-time Road Safety Monitoring Platform. `https://safety.transoftsolutions.com/trafxsafe-connect/`.

[368] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *EuroS&P*, pages 3–18. IEEE, 2017.

[369] T. Tsai, K. Yang, T.-Y. Ho, and Y. Jin. Robust Adversarial Objects against Deep Learning Models. In *AAAI*, 2020.

[370] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring Adversarial Robustness of Multi-Sensor Perception Systems in Self Driving. *arXiv:2101.06784*, 2021.

[371] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun. Physically Realizable Adversarial Examples for LiDAR Object Detection. In *CVPR*, 2020.

[372] Y. Tu, Z. Lin, I. Lee, and X. Hei. Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors. In *USENIX Security*, pages 1545–1562, 2018.

[373] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski. Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components. In *IV*, 2018.

[374] UK ACPO Road Policing Enforcement Technology Committee. *ACPO Code of Practice for Operational Use of Enforcement Equipment*. 2002.

[375] United States Department of Transportation - Federal Highway Administration. Manual on Uniform Traffic Control Devices for Streets and Highways - Chapter 2B. Regulatory Signs. `https://mutcd.fhwa.dot.gov/htm/2003r1/part2/part2b1.htm`.

[376] T. Urbanik, A. Tanaka, B. Lozner, E. Lindstrom, K. Lee, S. Quayle, S. Beaird, S. Tsoi, P. Ryus, D. Gettman, et al. *Signal Timing Manual*, volume 1. Transportation Research Board Washington, DC, 2015.

[377] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *CCS*, 2016.

[378] U.S. Department of Transportation (USDOT). Connected Vehicle Pilot Deployment Program Shares Open Source Cybersecurity Advances with Automakers. `https://www.its.dot.gov/pilots/cybersecurity_automakers.htm`.

[379] U.S. Department of Transportation (USDOT). Security Credential Management System (SCMS). `https://www.its.dot.gov/resources/scms.htm`.

[380] U.S. Department of Transportation (USDOT). Tampa (THEA) Connected Vehicle Pilots Works with the Infrastructure Integrator to improve Vehicle to Infrastructure (V2I) Connected Vehicle (CV) Applications. `https://www.its.dot.gov/pilots/thea_v2i_cv.htm`.

[381] U.S. Department of Transportation (USDOT). USDOT Connected Vehicle Pilot Deployment Program. `https://www.its.dot.gov/pilots/`.

[382] U.S. Department of Transportation (USDOT). Wyoming (WY) DOT Pilot. `https://www.its.dot.gov/pilots/pilots_wydot.htm`.

[383] Velodyne. Velodyne HDL-32E Datasheet. `https://velodynelidar.com`.

[384] Virginia Department of Transportation. Spacing Standards for Commercial Entrances, Signals, Intersections, and Crossovers. `https://www.virginiadot.org/projects/resources/access_management/12.27.11/Overview_of_Revised_Appendix_F_Spacing_Standards_12.2011.pdf`.

[385] J. Volpe. Vehicle-infrastructure integration (VII) initiative benefit-cost analysis Version 2.3. *United States Department of Transportation. Tech. Rep*, 2008.

[386] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song. Robust and Precise Vehicle Localization based on Multi-Sensor Fusion in Diverse City Scenes. In *ICRA*, pages 4670–4677. IEEE, 2018.

[387] D. Wang, C. Li, S. Wen, Q.-L. Han, S. Nepal, X. Zhang, and Y. Xiang. Daedalus: Breaking Nonmaximum Suppression in Object Detection via Adversarial Examples. *IEEE Transactions on Cybernetics*, 2021.

[388] J. Wang, A. Liu, Z. Yin, S. Liu, S. Tang, and X. Liu. Dual Attention Suppression Attack: Generate Adversarial Camouflage in Physical World. In *CVPR*, 2021.

[389] W. Wang, Y. Yao, X. Liu, X. Li, P. Hao, and T. Zhu. I Can See the Light: Attacks on Autonomous Vehicles Using Invisible Lights. In *ACM CCS*, 2021.

[390] Y. Wang, H. Xia, Y. Yao, and Y. Huang. Flying Eyes and Hidden Controllers: A Qualitative Study of People's Privacy Perceptions of Civilian Drones in The US. *Proc. Priv. Enhancing Technol.*, 2016.

[391] Z. Wang, W. Ren, and Q. Qiu. LaneNet: Real-Time Lane Detection Networks for Autonomous Driving. *arXiv:1807.01726*, 2018.

[392] D. Watzenig and M. Horn. *Automated Driving: Safer and More Efficient Future Driving*. Springer, 2016.

[393] The Waymo Driver Handbook: How our highly-detailed maps help unlock new locations for autonomous driving. `https://blog.waymo.com/2020/09/the-waymo-driver-handbook-mapping.html`.

[394] Waymo Self-Driving Car Gets Stuck by Cones, Drives Away From Assistance. `https://www.vice.com/en/article/y3dv55/waymo-self-driving-car-gets-stuck-by-cones-drives-away-from-assistance`.

[395] Waymo Safety Report 2021. `https://storage.googleapis.com/waymo-uploads/files/documents/safety/2021-08-waymo-safety-report.pdf`.

[396] R. Weston, O. P. Jones, and I. Posner. There and Back Again: Learning to Simulate Radar Data for Real-World Applications. In *ICRA*, pages 12809–12816. IEEE, 2021.

[397] W. Whyte, J. Petit, V. Kumar, J. Moring, and R. Roy. Threat and Countermeasures Analysis for WAVE Service Advertisement. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1061–1068. IEEE, 2015.

[398] X. Wu, H. X. Liu, and N. Geroliminis. An Empirical Analysis on the Arterial Fundamental Diagram. *Transportation Research Part B: Methodological*, 45(1):255–266, 2011.

[399] Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein. Making an Invisibility Cloak: Real World Adversarial Attacks on Object Detectors. In *ECCV*, 2020.

[400] C. Xiang and P. Mittal. DetectorGuard: Provably Securing Object Detectors against Localized Patch Hiding Attacks. In *ACM CCS*, 2021.

[401] C. Xiao, D. Yang, B. Li, J. Deng, and M. Liu. MeshAdv: Adversarial Meshes for Visual Recognition. In *CVPR*, 2019.

[402] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial Examples for Semantic Segmentation and Object Detection. In *ICCV*, 2017.

[403] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He. Feature Denoising for Improving Adversarial Robustness. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[404] H. Xu, Z. Tian, J. Wu, H. Liu, J. Zhao, et al. High-Resolution Micro Traffic Data from Roadside LiDAR Sensors for Connected-Vehicles and New Traffic Applications. Technical report, University of Nevada, Reno. Solaris University Transportation Center, 2018.

[405] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin. Adversarial T-shirt! Evading Person Detectors in A Physical World. In *ECCV*, 2020.

[406] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *arXiv:1704.01155*, 2017.

[407] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *NDSS*, 2018.

[408] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu. SoK: A Minimalist Approach to Formalizing Analog Sensor Security. In *IEEE S&P*, 2020.

[409] C. Yan, W. Xu, and J. Liu. Can You Trust Autonomous Vehicles: Contactless Attacks Against Sensors of Self-Driving Vehicle. *DEF CON*, 24, 2016.

[410] K. Yang, T. Tsai, H. Yu, M. Panoff, T.-Y. Ho, and Y. Jin. Robust Roadside Physical Adversarial Attack Against Deep Learning in Lidar Perception Modules. In *Asia CCS*, 2021.

[411] S. Yenikaya, G. Yenikaya, and E. Düven. Keeping the Vehicle on the Road - A Survey on On-Road Lane Detection Systems. *ACM Computing Surveys (CSUR)*, 46(1):1–43, 2013.

[412] C. You, Z. Hau, and S. Demetriou. Temporal Consistency Checks to Detect LiDAR Spoofing Attacks on Autonomous Vehicle Perception. In *MAISP*, 2021.

[413] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE access*, 8:58443–58469, 2020.

[414] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang. All Your GPS Are Belong To Us: Towards Stealthy Manipulation of Road Navigation Systems. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1527–1544, 2018.

[415] D. Zhang, J. Gabaldon, L. Lauderdale, M. Johnson-Roberson, L. J. Miller, K. Barton, and K. A. Shorter. Localization and Tracking of Uncontrollable Underwater Agents: Particle Filter Based Fusion of On-Body IMUs and Stationary Cameras. In *ICRA*. IEEE, 2019.

[416] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas. Real-Time Attack-Recovery for Cyber-Physical Systems Using Linear Approximations. In *2020 IEEE Real-Time Systems Symposium (RTSS)*, pages 205–217. IEEE, 2020.

[417] Y. Zhang, P. H. Foroosh, and B. Gong. CAMOU: Learning A Vehicle Camouflage For Physical Adversarial Attack On Object Detections In The Wild. *ICLR*, 2019.

[418] Z. Zhang, S. Liu, G. Tsai, H. Hu, C.-C. Chu, and F. Zheng. Pirvs: An Advanced Visual-Inertial SLAM System with Flexible Sensor Fusion and Hardware Co-design. In *ICRA*, pages 1–7. IEEE, 2018.

[419] D. Zhao, Y. Guo, and Y. J. Jia. Trafficnet: An Open Naturalistic Driving Scenario Library. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–8, 2017.

[420] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen. Seeing isn't Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors. In *ACM CCS*, 2019.

[421] Z. Zhong, W. Xu, Y. Jia, and T. Wei. Perception Deception: Physical Adversarial Attack Challenges and Tactics for DNN-Based Object Detection. In *Black Hat Europe*, 2018.

[422] H. Zhou, W. Li, Y. Zhu, Y. Zhang, B. Yu, L. Zhang, and C. Liu. Deepbillboard: Systematic Physical-World Testing of Autonomous Driving Systems. In *International Conference on Software Engineering*, 2020.

[423] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang. Online Multi-Object Tracking with Dual Matching Attention Networks. In *ECCV*, pages 366–382, 2018.

[424] X. Zhu, X. Li, J. Li, Z. Wang, and X. Hu. Fooling thermal infrared pedestrian detectors in real world using small bulbs. In *AAAI*, 2021.

[425] Y. Zhu, C. Miao, F. Hajiaghajani, M. Huai, L. Su, and C. Qiao. Adversarial Attacks against LiDAR Semantic Segmentation in Autonomous Driving. In *SenSys*, 2021.

[426] Y. Zhu, C. Miao, T. Zheng, F. Hajiaghajani, L. Su, and C. Qiao. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving? In *ACM CCS*, 2021.

[427] A. Zolfi, M. Kravchik, Y. Elovici, and A. Shabtai. The Translucent Patch: A Physical and Universal Attack on Object Detectors. In *CVPR*, 2021.

[428] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang. Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks. *IEEE Transactions on Vehicular Technology*, 2019.

[429] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang. LIC-Fusion: LiDAR-Inertial-Camera Odometry. *arXiv preprint arXiv:1909.04102*, 2019.

[430] X. Zuo, P. Geneva, Y. Yang, W. Ye, Y. Liu, and G. Huang. Visual-Inertial Localization With Prior LiDAR Map Constraints. *IEEE Robotics and Automation Letters*, 4(4):3394–3401, 2019.

Appendix A

Success Criteria of FusionRipper Attack

The required deviations under off-road and wrong-way attacks are calculated based on common widths of the AD vehicle, lane, and the road shoulder. These values differ in local and highway settings. Fig. A.1 shows the width measurements we used in the calculation. For the AD vehicle width, we use the width (including mirrors) of the Baidu Apollo's reference car, Lincoln MKZ [1]. For the lane widths and shoulder widths, we refer to the design guidelines [344] published by the US Department of Transportation Federal Highway Administration. For off-road attack, we use the deviation when the AD vehicle goes *beyond* the road shoulder from the center of the lane as the required deviation, which is calculated using $\frac{L-C}{2} + S = 0.895m$ (local) and $1.945m$ (highway), where L is the lane width, C is the car width, and S is the road shoulder width. For wrong-way attack, we define the required deviation as the AD completely invades the neighbor lane, and it is calculated with $\frac{L}{2} + \frac{C}{2} = 2.405m$ (local) and $2.855m$ (highway). We calculate the deviation of *touching the lane line* using $\frac{L-C}{2}$, which is $0.295m$ on local roads and $0.745m$ on the highway.

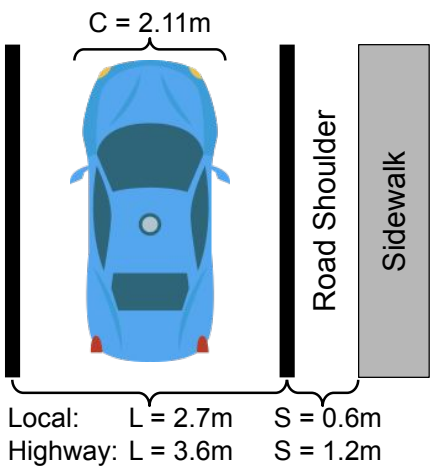

Figure A.1: Common AD vehicle, traffic lane, and road shoulder widths.

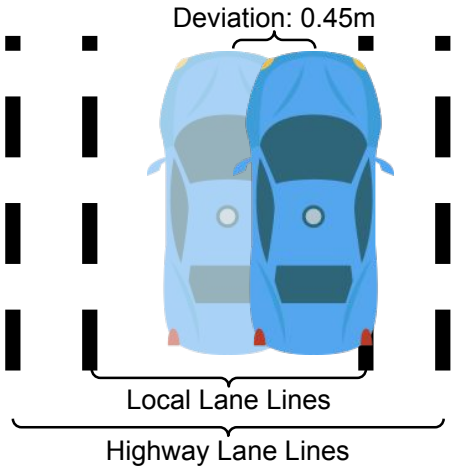

Figure A.2: Visualization of the lateral deviation 0.45 meters on local and highway roads.

Appendix B

Calculation of Lateral Position and Heading Rate Changes

Fig. B.1 shows the mathematical conversion from the steering angle to physical world lateral position change. The position change can be calculated as $\delta_{\text{pos}} = vt\sin(\frac{\theta}{\phi})$, where v is the velocity, t is the cycle time of the controller, θ is the steering angle, and ϕ is the steering ratio, which is a constant describing the ratio of the turning angle of the steering wheel to that of the vehicle wheel. The steering angle can be directly converted to heading rate change using $\delta_\omega = \theta/\phi t$, where δ_ω is the yaw (i.e., heading) rate change.

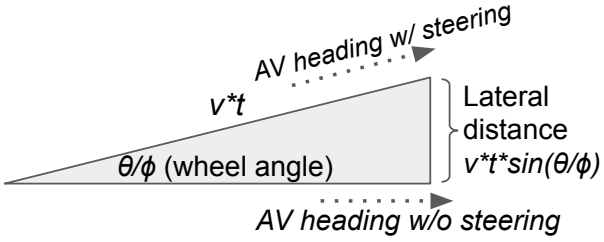

Figure B.1: Conversion from the steering wheel angle to lateral position change.

Appendix C

Success Criteria of DRP Attack

Required deviations. The required deviations for the highway and local roads are calculated based on Toyota RAV4 width (including mirrors) and standard lane widths in the U.S. [292] as shown in Fig. C.1. We use Toyota RAV4 since it is the reference vehicle used by the OpenPilot team when collecting the comma2k19 data set [329]. For the lane widths, we refer to the design guidelines [292] published by the U.S. Department of Transportation Federal Highway Administration. The required deviations to touch the lane line are calculated using $\frac{L-C}{2} = 0.735m$ (highway) and $0.285m$ (local), where L is the lane width and C is the vehicle width.

Required success time. Since ALC systems assume a fully attentive human driver who is prepared to take over at any moment [34, 326], the required deviation above needs to be achieved fast enough so that the human driver cannot react in time to take over and steer back. Thus, when we define the attack goal, we require not only the required deviation above, but also an attack success time that is smaller than the average driver reaction time to road hazards. We select the average driver reaction time based on different government-issued transportation policy guidelines [12, 374, 163, 134]. In particular, in the California

Department of Motor Vehicles Commercial Driver Handbook Section 2.6.1 [12], it describes (1) a 1.75 sec *average perception time*, i.e., the time from the time the driver's eyes see a hazard until the driver's brain recognizes it, and (2) a 0.75 to 1 sec *average reaction time*, i.e., the time from the driver's brain recognizing the hazard to physically take actions. Thus, in total it's **2.5 to 2.75 sec** from the driver's eyes seeing a hazard to physically take actions. The UK "Highway Code Book" and "Code of Practice for Operational Use of Road Policing Enforcement Technology" use 3 sec for driver reaction time [374, 163]. National Safety Council also adopts a 3-sec driver reaction time to calculate the minimum spacing between vehicles [134]. Among them, we select the **smallest** one, i.e., 2.5 sec from the California Department of Motor Vehicles [12], as the required success time in this paper to avoid possible overestimation of the attack effectiveness in our evaluation.

Note that the driver reaction time above is commonly referring to the reaction time to apply the brake, instead of steering. In our paper, we use such reaction time to apply the brake as the reaction time to take over the steering wheel when the ALC systems are in control of the steering wheel. This is because in traditional driving, the driver is *actively* steering the vehicle but *passively* applying the brake. However, when the ALC system is controlling the steering, the human driver is *passively* steering the vehicle, i.e., her hands are not actively controlling the steering wheel. Thus, the reaction time to take over the steering wheel during passive steering is analogous to that to apply the brake during passive braking.

In fact, the actual average driver reaction time when the ALC system is taking control is likely to be much higher than the 2.5 sec measured in traditional driving, due to the reliance of human drivers on such convenient driving automation technology today. A recent study performed a simulation-based user study on Tesla Autopilot, and found that 40% drivers fail to react in time to avoid a crash happening *6.2 sec* after the Autopilot fails to operate [252]. Thus, the required success time of 2.5 sec used in this paper is a relatively conservative estimation, and thus the attack effectiveness reported in our evaluations is likely only a

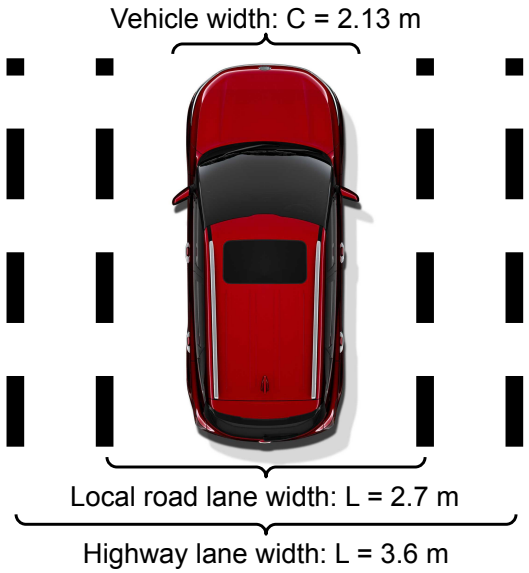

Figure C.1: Vehicle and lane widths used in this paper.

lower bound of the actual effectiveness of our attack in the real world.

Appendix D

Detailed DRP Attack Design

Differentiable construction of curve fitting (§4.2.3). Since the direct LD model output is the detected left and right lane line points (§2.1.1), a curve fitting step is further required to calculate $\rho_t(d; \{X_j^a | j \leq t\})$ in Eq. 4.8 from the lane line points. This step also needs to be differentiable to allow the entire $f(\cdot)$ differentiable with respect to $\{X_j^a | j \leq t\}$. Thus, we further perform a differentiable construction of the curve fitting process as follows. We use $P_l, P_r \in \mathbb{R}^{|D_t|}$ to represent the left and right lane line points respectively, where their indexes represent x-axis (longitudinal coordinate) and their values represent the y-axis (lateral coordinate). We first fit the lane line points into polynomial curves in a least-square manner with $\xi_l = (V^T V)^{-1} V^T P_l$ and $\xi_r = (V^T V)^{-1} V^T P_r$, where $\xi_l, \xi_r \in \mathbb{R}^{n+1}$ are the coefficients of the n-degree polynomial functions of the left and right lane lines respectively, and $V \in \mathbb{R}^{|D_t| \times n+1}$ is a Vandermonde matrix. Then we calculate the desired driving path coefficients ξ_d by averaging those for the left and right lane lines: $\xi_d = \frac{1}{2}(\xi_l + \xi_r)$. As all operations above are written in closed form, the desired driving path polynomial $\rho_t(d) = [1, d, d^2, ..., d^n]\xi_d$ is differentiable by each lane line point.

Gradient aggregation in BEV space (§4.2.3). In the gradient averaging step in Fig. 4.8

(step iii), we project the gradients w.r.t $X_1^a, ..., X_T^a$ into the BEV space, and then calculate the average value of them weighted by their corresponding visible patch area sizes in the model inputs. The weight is the number of pixels of the patch in the model input and normalized over all frames, i.e., the sum of the weights equals 1. This weighted averaging is designed to prevent the averaged gradient from being dominated by the earlier frames, where the patch is far and small but the whole patch is visible.

Note that this procedure does not produce the true gradient on $\text{BEV}^{-1}(X_i^a)$. Instead, this is approximation of this true gradient to avoid engineering efforts in deriving the differentiation of $\text{BEV}^{-1}(\cdot)$ code in OpenCV. This also allows us to control the aggregation weights more flexibly. $\text{BEV}(\cdot)$ consists of matrix-vector multiplication and scaling. This approximation works since in our case $\text{BEV}(\cdot)$ and $\text{BEV}^{-1}(\cdot)$ are close to a linear transformation as the scaling is not substantially different across consecutive frames.

Appendix E

Detailed LD^3 Design and Implementation Choices

Converting LD outputs to lateral deviations. The LD output consists of the detected left and right lane lines, which are represented as polynomial functions in the bird's eye view [7, 131]. An example of the polynomial functions is shown in Fig. E.1. For these polynomial functions, the absolute values at $x = 0$ represent the vehicle's distances to the lane lines, d_{left} and d_{right}. Therefore, we can calculate the lateral deviation to the lane centerline by

$$lw/2 - d_{\text{left}} \quad \text{or} \quad d_{\text{right}} - lw/2, \tag{E.1}$$

where lw is the lane width. We calculate the lateral deviation as a *signed* number to differentiate the deviations to the left (positive) and to the right (negative).

Although we can also obtain the lane width from the lane line polynomials (i.e., $lw_{\text{poly}} = d_{\text{left}} + d_{\text{right}}$), as mentioned in §5.4.3, it is not uncommon that one of the lane lines is missing or incorrectly detected in real world driving, e.g., when the current lane splits into a through lane

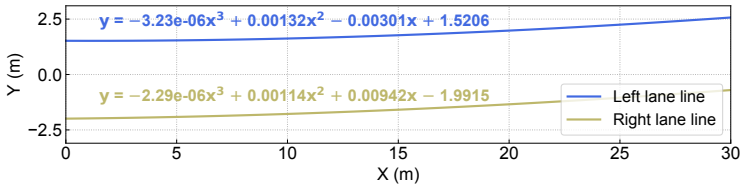

Figure E.1: Example of left/right lane line polynomial functions.

Algorithm 5 Calculation of LD deviation to lane centerline

Notations: lw_{map}: lane width from map; $poly(\cdot)$: polynomial function fitted on detected lane line; d: distance to lane line; D: deviation to lane centerline

1: **function** LDDEV(LD, lw_{map})
2: $d_{\text{left}} \leftarrow |LD.poly_{\text{left}}(0)|$ **if** $LD.poly_{\text{left}}$ **else** ∞ ▷ dist. to left line
3: $d_{\text{right}} \leftarrow |LD.poly_{\text{right}}(0)|$ **if** $LD.poly_{\text{right}}$ **else** ∞ ▷ dist. to right line
4: **if** $LD.poly_{\text{left}}$ **and** $d_{\text{left}} < d_{\text{right}}$ **then** ▷ left line is correct
5: $D \leftarrow lw_{\text{map}}/2 - d_{\text{left}}$ ▷ dev. to centerline; +: left, −: right
6: **else if** $LD.poly_{\text{right}}$ **and** $d_{\text{right}} < d_{\text{left}}$ **then** ▷ right line is correct
7: $D \leftarrow d_{\text{right}} - lw_{\text{map}}/2$
8: **else** ▷ if none of the lane lines are correctly detected
9: $D \leftarrow$ last calculated D ▷ re-use last dev. to centerline
10: **end if**
11: **return** D
12: **end function**

and a left or right turn lane. In such cases, directly using the distances from the polynomial functions would result into a wrong lateral deviation. To address this, we include two optimizations in the lateral deviation calculation: (1) instead of estimating the lane width from the polynomial functions, we query the current lane width from the semantic map (line 4 in Alg. 2), and (2) prioritize the lane line with a smaller distance to the vehicle by using it to calculate the lateral deviation in Eq. E.1 (line 4, 6 in Alg. 5). This is because for the lane splitting scenario mentioned above, the incorrectly-detected lane line often has a much larger distance compared to the correctly-detected one. A special handling is that when both lane lines are incorrectly detected, which is very rare in SCNN [297] and never occur in OpenPilot LD model [131], we will reuse the previously calculated lateral deviation.

Safe deceleration in attack response. Generally, a deceleration <4.6 m/s^2 is considered as safe for maintaining steady control [308]. Thus, to calculate the speed profile of the AR trajectory (§5.4.4), we apply 4 m/s^2 as the deceleration, which is also defined in Baidu Apollo as the maximum allowed deceleration to ensure safety [7].

Appendix F

Independence between LiDAR Localization and Lane Line Markings

Evaluation methodology. To evaluate the dependency of LiDAR localization on lane line markings, we first create two traces of modified LiDAR data: one without lane line markings (denote as *no-marking*) and another with incorrect lane line markings (denote as *wrong-marking*). Next, we execute the LiDAR locators on the original LiDAR trace as well as on the two modified traces. *If a LiDAR locator does not rely on the lane line markings, we should observe a high similarity between the original and the modified executions.*

Specifically, LiDARs scan the surrounding environment and output Point Cloud Data (PCD), which stores the 3D positions and intensities of the reflected laser points. Since the lane line markings will exhibit distinctively higher intensities than the other road surface due to their color differences, we create the *no-marking* PCDs by changing their intensities to the same as other road surface area. To do that, we first apply the commonly-used RANSAC plane segmentation [330] on the PCDs to find all points that belong to the ground plane, i.e., the road surface, and then set the intensities of these ground points to their median value. This

thus effectively makes the lane line markings indistinguishable from the other road surface. The creation of *wrong-marking* PCDs is slightly more complicated. After recognizing all ground points, we identify the lane line marking points depending on whether their intensities are above a certain threshold. For each lane line marking point, we search a corresponding ground point that is *laterally offset by half-lane-width* and set their intensities the same as the original lane line marking points. Finally, we clear the original lane line marking points by setting their intensities to the median ground point intensities. Since the lane line markings are moved by half-lane-width, the *wrong-marking* PCDs should have the largest lateral LiDAR localization impact if the lane line markings have any effect on the LiDAR locator. Fig. F.1 shows such an example of the original PCD and the one with *no-marking* and *wrong-marking*.

Experimental setup. We evaluate on 2 LiDAR locators, one from Baidu Apollo (BA-LiDAR locator) [386] and another from Autoware (AW-LiDAR locator) [213]. At design level, BA-LiDAR locator considers point cloud intensities in its position calculation. Thus, the modifications of lane line intensities do have the potential to affect the BA-LiDAR locator performance. On the other hand, AW-LiDAR locator only uses the position data in the PCD and completely ignores the intensities. This means that AW-LiDAR locator does not consider lane line markings at the design level. Since AW-LiDAR locator implements the Normal Distributions Transform (NDT) algorithm [102], which does not output position uncertainty by default, thus we follow a common adaptation for NDT to use the point cloud matching fitness score as the uncertainty [270].

Since MSF localization takes not only position measurements but also position uncertainties from LiDAR locator as inputs, we calculate both the *position accuracies* and *uncertainty correlation* with the original and no/wrong-marking PCDs to show the similarity. We evaluated on the same 5 local road and highway traces in *FusionRipper* [335] from two datasets. For each trace, we exclude intersections since they do not have lane line markings. Among them,

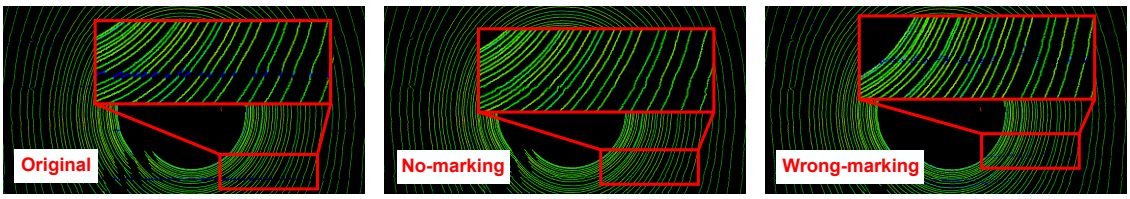

Figure F.1: PCDs with the original, removed, and incorrect lane line markings.

Table F.1: The uncertainty correlation coefficients (r) and position accuracies (RMSE) of LiDAR locators using the original, no-marking, and wrong-marking. Results with statistically strong correlation are highlighted in **bold** (p-values are all statistically significant).

	Uncertainty Correlation (r)		Position Accuracy (RMSE)		
	Original vs No-marking	Original vs Wrong-marking	Original	No-marking	Wrong-marking
BA-LiDAR Locator	**0.89**	**0.64**	0.064 m	0.065 m	0.063 m
AW-LiDAR Locator	**1.0**	**1.0**	0.076 m	0.076 m	0.076 m

since *ba-local* does not provide ground truth positions, we calculate the position accuracy based on the LiDAR locator with the original lane line markings.

Results. Table F.1 shows the experiment results. For the position accuracy, we report the Root Mean Squared Error (RMSE) between LiDAR locator positions and ground truth positions or the ones with original lane line markings. For the correlation, we use the commonly-used Pearson's correlation, and a correlation coefficient >0.5 is considered strongly correlated [128]. As shown, for both LiDAR locators, the uncertainty correlation coefficients between the original and modified PCDs are all well above the threshold for strong correlation, and their position accuracies are also all at centimeter-level. Particularly, since AW-LiDAR locator does not use lane line markings at the design level, the traces consequently show perfect correlations and identical position accuracies no matter how we modify the lane line markings. Such a result suggests that the existing LiDAR locators used in high-level AD systems are indeed largely ignore the lane line marking information when localizing the vehicle on the map, which might because global localization focuses more on the unique features on the road, such as buildings, roadside layouts, and traffic signs. As a result, this indicates that lane line markings are largely independent of the ones that are already used in high-level AD localization and thus pose a great potential for defense purposes.

Appendix G

SAVIOR Evaluation Details

SAVIOR Evaluation Setup

To evaluate SAVIOR, we follow the similar methodology as the ground rover evaluation in the SAVIOR paper [313], i.e., using the kinematic bicycle model [225] and an Extended Kalman Filter (EKF) to predict the system state (i.e., position in x, y coordinates) given the vehicle control commands (i.e., steering and acceleration). Although the vanilla bicycle model does not have tunable parameters, we follow a similar implementation as SAVIOR by adding coefficients to the bicycle model equations [26]. Same as SAVIOR, we use the *nlgreyest* system identification tool from Matlab [267] to find the coefficients that can best fit the sensor and control trace. During the evaluation, we continuously calculate the residuals between the GPS measurements and the predicted positions from the EKF, and feed the residuals to a CUSUM anomaly detector for attack detection. An execution that triggers the CUSUM detector will be considered as under attack.

Since the KAIST dataset [203] does not store the control commands when the traces were collected, which are required for the SAVIOR evaluation, we replay the KAIST sensor traces

as inputs to Baidu Apollo v5.0.0 [7] to collect the control module outputs, i.e., steering and throttle commands. In particular, the control module calculates such commands based on the localization and a planned trajectory, which is a sequence of trajectory points that the vehicle should follow. However, the planned trajectory is runtime information optimized by the planning module during driving, which is not available in the dataset. Since the ground truth positions in the KAIST traces represent the trajectory points followed by the AD vehicle, we thus convert the ground truth positions into planned trajectories according to the format in Baidu Apollo and use them as one of the control inputs. With the planned trajectories, we then feed the benign localization and attacked localization outputs to obtain the benign control commands and attack influenced control commands respectively.

In addition to the KAIST traces, we also evaluate SAVIOR on a dataset that contains the original control commands to validate SAVIOR's detection performance in an ideal setting. However, similar performance is observed in that dataset to the ones on KAIST traces. More details of this are in Appendix G.

Evaluating SAVIOR on Dataset with Control Commands

Since SAVIOR requires vehicle control commands, for which we collected by replaying the KAIST traces in Baidu Apollo in our evaluation (Appendix G), one might argue that SAVIOR may perform much better if given the originally collected vehicle control commands. Therefore, we evaluate SAVIOR on the comma2k19 dataset [329], which contains the original vehicle control commands when the traces were collected. Since the comma2k19 dataset does not provide LiDAR data, we thus cannot run the MSF attack. To evaluate SAVIOR, we apply the most aggressive GPS spoofing parameters in the MSF attack ($d = 2.0$, $f = 2.0$) to the GPS data and examine SAVIOR's capability at detecting such obvious GPS spoofing attempts. As shown in Fig. G.1, SAVIOR's detection performance is close to the one on the

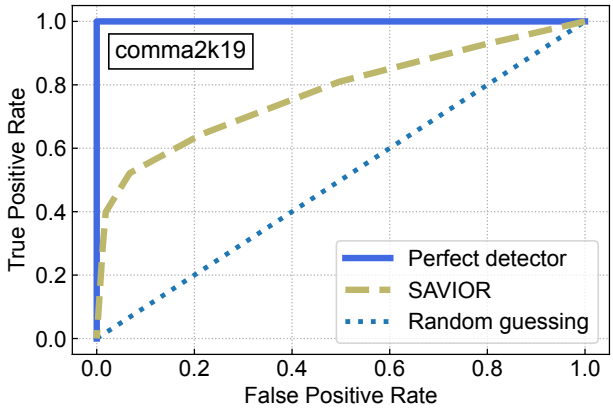

Figure G.1: Attack detection ROC curve of SAVIOR on comma2k19 [329].

ka-highway36 (Fig. 5.4) and is still far from a perfect detector.

Appendix H

AI Components Studied in Semantic AD AI Security Research

Perception

Road object detection identifies obstacles (e.g., bounding box), lane lines, and traffic lights from sensor data. The ones targeted by existing semantic AD AI security works includes:

Object detection and *segmentation* are commonly used to detect vehicles, pedestrians, cyclists, traffic objects (e.g., traffic cones), etc., in AD systems. Existing attacks on these aim to cause adversarial effects including *hiding* (i.e., making objects disappear), *creation* (i.e., detecting non-existing objects), and *alteration* (i.e., changing the types of objects).

Lane detection can detect lane line shapes and relative position in lane and is widely used in L2 AD systems for Automated Lane Centering. Prior works [327, 210] show lane detection models are vulnerable to physical perturbations on road, which can lead AD vehicle to drive out of lane line.

Traffic light detection identifies the traffic light and detects the current signal color. Prior works [355, 253, 389] leverages its dependency on localization or light projection to induce wrong detection, no detection, or color change.

MSF perception is commonly used in L4 AD systems [7] to fuse detection results from different perception sources (e.g., LiDAR and camera) for higher accuracy and robustness [167, 244, 122]. Prior works found that MSF perception is vulnerable to adversarial 3D objects [110, 370]. Such attacks can hide the adversarial objects or cause other obstacles being misdetected.

Object tracking builds the trajectories of one or more detected objects in a sensor frame sequence to tolerant the occasional false positions and false negatives. However, prior works [143, 121, 209] demonstrate successful adversarial attacks that can move a roadside vehicle into the driving lane or move a front vehicle out of the road [209, 205].

Localization

LiDAR/Camera localizations are commonly used in AD systems for localizing the vehicle. They operate by finding the best matched location of the live LiDAR point cloud or camera image in a map [102, 277]. Prior works discover that such localization algorithms may lead the sensitive location data [257] or can be disrupted by IR lights [389].

MSF localization is predominantly used in L4 AD systems to achieve robust and accurate localization by fusing location sources such as GPS, LiDAR, and IMU [386]. Prior work [335] finds that Kalman Filter based MSF design is vulnerable to strategic attacks leveraging a single attack source, such as GPS spoofing, to inject large deviations in the localization results.

Chassis

The Chassis component acts as the information hub of the vehicle, which periodically broadcast privacy-sensitive diagnosis information, including Vehicle Identification Number (VIN), to other components. Prior work [191] demonstrates using a compromised Robot Operating System (ROS) node to intercept the Chassis message to steal the VIN.

End-to-End Driving

End-to-End driving [353] is a distinctive AD system design paradigm from the more common modular designs used in industry-grade AD systems [7, 213]. Due to DNN-based design, end-to-end driving is vulnerable to adversarial attacks, which can cause the driving model to predict incorrect control commands [226, 181]. Prior work [251] show that trojan attack enables triggering specific driving behaviors (e.g., turn right) using road signs with specific patterns.

Appendix I

STOP Sign Attack Reproduction

ShapeShifter (SS) [120] reproduction and results. We directly use the official open-source code [30] to reproduce SS on Faster R-CNN [321]. With that, we can ensure that the attack generated by the source code should have the same effect as the original work. We validated in simulation using the same distance/angle settings as in the paper. Results show that our reproduction can achieve 80% (12/15) attack success rate while the original paper reports 73% (11/15). In addition, the successful attack distances/angles of the reproduced attack align well with the paper with the only exception that our reproduction is successful at 25 feet / 0° but the paper reports failure. Such results also indicate a sufficient simulation fidelity, which is consistent with our fidelity evaluation in §7.5.4.

Robust Physical Perturbation (RP2) [157] reproduction and results. Same as the paper, we select YOLOv2 [318] as the targeted object detection model. Specifically, we implement the loss function including adversarial loss with Expectation over Transformation [90], total variation loss, l_p loss, and non-printability score loss [333] as used in RP2 design [157]. We compare the attack effectiveness in simulation to the reported results in the paper in the same distance settings (0–30 feet in outdoor environment). Our results show

that we can achieve an attack success rate of 66.4% in all camera frames, which is similar to the 63.5% reported in the paper.

Seeing Isn't Believing (SIB) [420] reproduction/modeling. Since SIB is not open-sourced, we tried our best to reproduce it but were still unable to achieve the same attack effectiveness shown in their paper [420]. Nevertheless, in our simulation platform evaluation (§7.5.3), we apply SIB using a modeling-based approach, i.e., by sampling STOP sign detection failures using *exactly the same* frame-wise attack success rate at different STOP sign distances/angles reported in the paper. Specifically, the attack success rates in the paper are from 32–94% for distances from 5–25 m on YOLOv3 [420].

Appendix J

Example AD and Plant Model Setups in *PASS*

AD model setup. We illustrate the detailed AD model setup for the STOP sign attack evaluation (§7.5.3) as an example to demonstrate the usage of *PASS*. Specifically, we adopt the typical modular AD pipeline design including object detection, object tracking, fusion, planning, and control.

Object detection. We use the same models and detection thresholds as originally used in the STOP sign attacks: SS on Faster R-CNN (threshold: 0.3) [120], RP2 on YOLOv2 (threshold: 0.4) [157], and SIB on YOLOv3 (threshold: 0.4) [420].

Object tracking. We adopt a general Kalman Filter based multi-object tracker [259] which tracks the STOP sign locations and sizes. Next, we convert the detected STOP sign location from 2D image coordinates to real-world coordinates. Here, we adopt 2 variants: (1) HD Map based (denoted *Map*). Similar to the design in Tesla Autopilot [360], we use the GPS position to query the HD Map for the distance to the front intersection and use it as an approximation for the STOP sign distance. (2) Pinhole camera based (denoted *Pinhole*). We

apply the pinhole camera model to estimate the STOP sign distance based on the heuristics on standard STOP sign sizes [375], which is a similar design to Apollo [84]. For track management, a new STOP sign track is created when the STOP sign is detected for $0.2 \times \text{FPS} = 4$ (frames per second) frames, and an existing track will be deleted after misdetection for $2 \times \text{FPS} = 40$ consecutive frames (parameters recommended by Zhu et al. [423]).

Fusion. The fusion component is optional and thus can be disabled in the pipeline. If enabled, it can fuse object detection results from multiple data sources. In addition, if the HD Map contains detailed static road objects (e.g., traffic signs) it can serve as a data source to fuse with object detection model outputs. Since the output format of all fusion sources are consistent (i.e., objects), we extend the object tracker to accept detections from multiple sources to facilitate the fusion.

Planning. We adopt a lane following planner where the future trajectory locates at the center of current driving lane [7]. The planner by default maintains a desired speed, but reduces it if *a STOP sign presents* or *there exists a front obstacle with a close distance or slowing down*. Taking the STOP sign as an example, we first calculate a *braking distance* based on the current driving speed and the safe vehicle deceleration ($-3.4m/s^2$ [308]). If a STOP sign is currently tracked and its distance to the vehicle reaches the braking distance, the planner starts to decelerate. If the STOP sign is no longer tracked and current speed is smaller than the desired driving speed (i.e., the speed that the vehicle intends to maintain if no STOP sign), the planner accelerates the vehicle.

Control. The control module will execute the planning decisions, and we use the classic Stanley controller [187] for lateral control (i.e., steering) and a Proportional-Integral-Derivative (PID) controller [82] for longitudinal control (i.e., throttling and braking). Consistent with existing L2 AD systems such as OpenPilot [294], We set the frequency of detection, tracking, fusion, and planning as 20 Hz, and control as 100 Hz.

In our evaluation (§7.5.3), We adopt 3 AD pipeline variants based on the availability of HD Map and fusion: *Map*, *Pinhole*, and *Fusion*. Specifically, *Map* and *Pinhole* vary in the STOP distance estimation, and *Fusion* refers to fusing STOP sign detection with the one from the HD Map.

Plant model setups. *PASS* supports two options for the plant model. For simulation-based evaluation, we adopt the SVL [238], which is an industry-grade AD simulator. For real-world experiments, we can use the two L4-capable AD vehicles in our project team as shown in Fig. 7.3, where they are equipped with AD-grade sensors including multiple short- and long-range cameras, LiDARs (16-line, 32-line, and 64-line), mmWave RADAR, duel-antennas GPS with RTK, high-precision IMU, etc. As described in §7.5.1, the platform is simulation-centric; it is mainly for simulation-based evaluation and only use the real vehicles for simulation fidelity improvements and physical world evaluation in exceptional cases.