Attacks and Defenses on Autonomous Vehicles: From Sensor Perception to Control Area Networks

ATTACKS AND DEFENSES ON AUTONOMOUS VEHICLES: FROM SENSOR PERCEPTION TO CONTROL AREA NETWORKS

by

Yanmao Man

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2022

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Yanmao Man, titled Attacks and Defenses on autonomous Vehicles: From Sensor Perception to Control Area Networks and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Ming Li — Date: July 18, 2022

Loukas Lazos — Date: July 18, 2022

Gregory Ditzler — Date: July 18, 2022

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Ming Li — Date: July 18, 2022
Dissertation Committee Chair
Electrical and Computer Engineering

ACKNOWLEDGMENTS

First, I would like to give my most sincere thanks to my advisor, Dr. Ming Li. Without his guidance and support, I could've been the same person as I was five years ago, whereas now I can proudly say that I earned my Ph.D. degree from the University of Arizona and my advisor is Dr. Li. Yes, "is" because I believe the advice I received from Dr. Li is not just for those five years but can last for a lifetime. He taught me how to think, research, present, and many more. Perhaps most importantly, he taught me all of these with patience, a level of patience that I don't even think I could have for myself, so thank you, Dr. Li.

I would like to express my special gratitude to Prof. Ryan Gerdes from Virginia Tech, who has not only provided a great amount of ingenious research advice as if she were my advisor, but also made my entire five-year journey more enjoyable.

Thanks also go to Dr. Loukas Lazos, Dr. Gregory Ditzler, and Dr. Christian Collberg, for serving on my comprehensive exam and/or dissertation defense committees. Dr. Collberg was also my minor advisor in Computer Science. Their insightful comments and suggestions greatly improved my dissertation and research.

I also want to thank my collaborators, Raymond Muller and Prof. Z. Berkay Celik from Purdue University, Mahsa Foruhandeh and Abdullah Zubair from Virginia Tech, and Mingshun Sun. Additionally, I thank all my labmates and colleagues for their help and friendship, including Ahmed Salama, Yanjun Pan, Tianchi Zhao, Mohamed Seif, Bo Jiang, Zhiwu Guo, Xiaolan Gu, Jingcheng Li, Ziqi Xu, and Zhengzhong Liang.

My deepest thanks go to my parents and my beloved wife, Luhong Wu. I am grateful that Luhong's been always there, sharing every moment of happiness or depression, but the sweetest thing is, she's there when it's just a regular Saturday afternoon, with Tikka, our cat!

Finally, I thank the financial support from NSF grants CNS-1801402, CNS-1410000, ARO grant W911NF2110320, and the teaching assistantships from the ECE department.

LAND ACKNOWLEDGMENT

We respectfully acknowledge the University of Arizona is on the land and territories of Indigenous peoples. Today, Arizona is home to 22 federally recognized tribes, with Tucson being home to the O'odham and the Yaqui. Committed to diversity and inclusion, the University strives to build sustainable relationships with sovereign Native Nations and Indigenous communities through education offerings, partnerships, and community service.

DEDICATION

To my parents and my wife

Contents

List of Figures

List of Tables

ABSTRACT

Autonomous driving has been a focus in both industry and academia. The autonomous vehicle decision-making pipeline is typically comprised of several modules from perceiving the physical world to interacting with it. The perception module aims to understand the surrounding environment such as obstacles, lanes, traffic signals, etc. This high-level information is passed to the planning module which generates decisions such as acceleration, turning right, etc. These decisions are made also based on a prediction module that estimates the future trajectories of obstacles for precaution purposes. The control module translates the decisions into low-level instructions, transmitted on the control area network (CAN) bus in the form of CAN messages. Finally, the actuation module executes these instructions.

Since autonomous vehicles normally operate at high speed and usually carry humans, their safety and security are of great importance. Recently, a number of papers raise the security concerns with various attacks. Since the perception module leverages deep learning models for object detection, traffic sign recognition, etc., it is inherently subject to adversarial examples that are specially crafted to deceive neural networks. More severely, those physically-realizable adversarial examples/patches that can be implemented externally using printed stickers or projectors, can bypass the digital protection of the autonomous systems thus being more practical, more stealthy, and more difficult to defend against. On the other hand, due to the lack of secure authentication, the CAN protocol has been demonstrated to be susceptible to ECU (electronic control unit) impersonation attacks where an attack-compromised ECU can broadcast forged CAN messages in order for dangerous actuation, such as deploying airbags on a highway even under normal driving circumstances.

In this dissertation, we study the security of autonomous vehicles from sensor perception to the CAN bus. We propose attacks that nullify a broad category of state-of-the-art (SOTA) defenses, and develop our own defenses that can be generalized to defeat different attack

methodologies. In particular, for the perception module we develop a visible light-based, system-aware camera attack, termed GhostImage that can be realized physically and remotely (Chapter 3). We exploit the ghost effect of the camera system to convey adversarial noise that is not norm-bounded, thus bypassing SOTA adversarial example defenses. To detect perception attacks, we propose to adopt the idea of spatio-temporal consistency, which is demonstrated using two different methods: one is model-based (Chapter 4) for detecting ghost-based camera attacks, and the other is data-driven (Chapter 5) in that we can detect object misclassification attacks effectively and efficiently, meanwhile our algorithm is agnostic to different attack methodologies as well as different object detection and tracking systems. In Chapter 6, we investigate the planning module and enhance the adversarial robustness of the obstacle trajectory prediction. Finally in Chapter 7, to evaluate the control and actuation modules we propose a Hill-climbing-style attack that defeats SOTA CAN bus intrusion detection systems that are based on multiple CAN frames.

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous vehicles have been under rapid development [14–16] with real-world deployments on the road [17]. Their systems are typically operated based on a pipeline where at one end it perceives the physical environment, while at the other end it interacts with the environment (Fig. 1.1). Particularly, in the perception module sensors such as (multiple) cameras and/or lidars are used to scan the surrounding, for which machine learning algorithms are used to analyze the raw sensory data in order to understand the scene at a higher level, e.g., what/where the obstacles are. These high-level information are then used by the prediction module to predict how these obstacles are going to move for precaution purposes [18]. Then, the planning module makes decisions of the reactions that the system should take, e.g., to decelerate for the pedestrian who is about to cross the street. These decisions are translated by the control module to low-level commands for the actuation module to execute. The commands are transmitted over the control area network (CAN bus) to which electronic control units (ECU) are connected.

Although people are optimistic about large-scale deployment of such systems in the near future, recent work has pointed out that they are however vulnerable to various attacks. The general attack goal is to jeopardize the victim vehicle's behavior which may result in severe traffic accidents. Based on the methodology, these attacks can be mainly categorized into external attacks and internal attacks.

External attacks refer to those physically-realizable perception attacks that inject false data into the perception which eventually affects the vehicle's actuation. These attacks compromise either the sensors themselves [7,25,26], or the physical environment/objects [21,22].

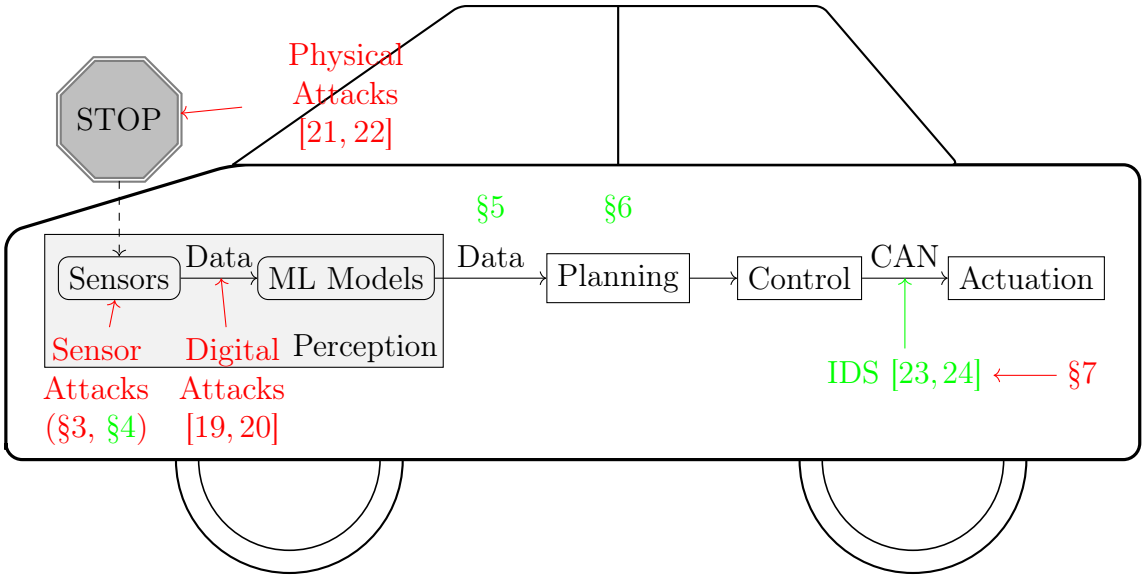


Figure 1.1: Autonomous vehicle systems: The security and the dissertation scope.

Cameras are one of the most common sensors, but existing *remote* attacks against cameras are limited to, essentially, denial-of-service attacks [25–27], which are easily detectable (e.g., by tampering detection [28]). These perception attacks alter the data at the source, hence bypassing traditional digital defenses (such as crypto-based authentication or access control), and are subsequently much harder to defend against [29, 30]. Most existing defenses against these attacks are specific to either the sensing modality (e.g., LiDAR [31], GPS [32], IMU [33]), or one particular attack methodology, e.g., norm-bounded attacks [34] or adversarial patch attacks [35]. Viable countermeasures against perception attacks are mostly based on verifying the consistency either among multiple sensors [33,36,37], or using empirical/semantic knowledge [38] such as optical flow [34] from a single camera.

Internal attacks, where the attacker gains access to the internal decision-making pipeline and alter the data digitally, include digital perception attacks, CAN bus attacks. In a digital perception attack, the attacker directly alters the sensor output (e.g., image pixels, or lidar points) such that the machine learning (ML) models (e.g., deep neural networks, DNNs) make incorrect predictions [19, 20]. Defenses against them often assume norm-bounded perturbation [39–41] thus fail to generalize.

In a CAN bus attack, the attacker tampers one or more ECUs either via local manners [42] or even remotely [43], and reprograms them to impersonate other ECUs, broadcasting CAN

frames that contain unauthorized actuation commands. Defenses against them mainly focus on physical layer identification, a type of authentication technique that identifies the message source by fingerprinting the characteristic of the message's physical signal [23, 24, 44]. A CAN frame can only accommodate eight bytes of both data and cryptographic information, hence, providing authentication and integrity via message authentication codes (MACs) or digital signatures is not a straightforward proposition. Existing solutions to the lack of authentication for CAN messages are either insecure or computationally intensive which makes them incapable of a reliable authentication on a frame-by-frame basis [45].

Given these existing attacks against autonomous vehicles as well as their countermeasures, in this dissertation we raise these two questions: *Are the state-of-the-art defenses secure?* If not, *how do we propose to safeguard the system?* However, it is non-trivial to answer them. For the first question, we need to design one type of attack that is general enough to defeat as many defenses as possible instead of just a single defense, because otherwise the security analysis of existing defenses is only opportunistic. To achieve that we need to unveil a common flaw of multiple, yet seemingly unrelated defenses, which is indeed challenging. If we found such attacks, we would need to answer the second question in a similar way; that is, we need to design defenses that overcome not only the attacks we found, but also those attacks with different methodologies so long as they share the same attack goal. In other words, our defenses need to be agnostic to attack methodologies (in order to avoid endless arm race where a new attack method demands a brand new defense and vice versa), which is undoubtedly challenging.

1.2 Main Contributions

In this dissertation, we present our first attempts to resolve these challenges.

1.2.1 Remote Perception Attacks against Camera-based Object Recognition Systems

In vision-based object classification systems imaging sensors perceive the environment and then objects are detected and classified for decision-making purposes; e.g., to maneuver an automated vehicle around an obstacle or to raise an alarm to indicate the presence of an

intruder in surveillance settings. In Chapter 3 we demonstrate how the perception domain can be remotely and unobtrusively exploited to enable an attacker to create spurious objects or alter an existing object. An automated system relying on a detection/classification framework subject to our attack could be made to undertake actions with catastrophic results due to attacker-induced misperception. We focus on camera-based systems and show that it is possible to remotely project adversarial patterns into camera systems by exploiting two common effects in optical imaging systems, viz., lens flare/ghost effects and auto-exposure control. To improve the robustness of the attack to channel effects, we generate optimal patterns by integrating adversarial machine learning techniques with a trained end-to-end channel model. We experimentally demonstrate our attacks using a low-cost projector, on three different image datasets, in indoor and outdoor environments, and with three different cameras. Experimental results show that, depending on the projector-camera distance, attack success rates can reach as high as 100% and under targeted conditions.

Our contributions are further summarized as follows.

- We are the first to study remote perception attacks against camera-based classification systems, whereby the attacker induces misclassification of objects by injecting light, conveying adversarially generated patterns, into the camera.

- Our attack leverages optical effects/techniques, namely, lens flare and auto-exposure control, that are widespread and common, making the attack likely to be effective against most cameras. Furthermore, we incorporate these effects in an end-to-end manner into an adversarial machine learning-based optimization framework to find the optimal patterns an attacker should inject to cause misperception.

- We demonstrate the efficacy of the attacks through experiments with varying image datasets, cameras, distances, and indoor to outdoor environments. Results show that GhostImage attacks are able to achieve attack success rates as high as 100%, depending on the projector-camera distance.

1.2.2 A Model-based Countermeasure for Object Detection Systems against Flare-based Perception Attacks

To defend against the aforementioned ghost-based camera attacks, we propose a model-based detection algorithm that is based on spatiotemporal consistency and physical rules. In particular, we develop a countermeasure that reduces the problem of detecting ghost-based attacks into verifying whether there is a ghost overlapping with a detected object. This is based on the fact that a successful attack must introduce such overlapping. We leverage spatiotemporal consistency to eliminate false positives. That is, if for both frames there are ghost-object overlaps, the algorithm can confidently claim that there is an attack. Evaluation on experimental data provides a worst-case equal error rate of 5%.

Our contributions are further summarized as follows.

- We propose to use spatiotemporal consistency to detect adversarial ghosts. The detection algorithms uncovers the attack based on the physical rules rather than a secret therefore being infeasible for an adaptive attacker to bypass.
- Our algorithm allows for attack prevention because it requires only two frames for detection. It can also detect attacks that do not succeed, i.e., a failed attack attempt.
- Security analysis establishes the robustness of our algorithm. Evaluation with real-world data further shows that our detection algorithms achieves a worst-case equal error rate of 5% at most, demonstrating high efficacy in protecting the autonomous system from ghost-based perception attacks.

1.2.3 Object Misclassification Attack Detection Using Data-Driven Spatiotemporal Consistency Verification

Autonomous systems commonly rely on object detection and tracking (ODT) to perceive the environment and predict the trajectory of surrounding objects for planning purposes. An ODT's output contains object classes and tracks that are traditionally predicted independently. Recent studies have shown that ODT's output can be falsified by various perception attacks with well-crafted noise, but existing defenses are limited to specific noise injection methods

and thus fail to generalize. In Chapter 5 we propose PercepGuard for the detection of misclassification attacks against perception modules regardless of attack methodologies. PercepGuard exploits the spatiotemporal properties of a detected object (inherent in the tracks), and cross-checks the consistency between the track and class predictions. To improve adversarial robustness against defense-aware (adaptive) attacks, we additionally consider context data (such as ego-vehicle velocity) for con textual consistency verification, which dramatically increases the attack difficulty. Evaluations with both real-world and simulated datasets produce a FPR of 5% and a TPR of 99% against adaptive attacks. A baseline comparison confirms the advantage of leveraging temporal features. Real-world experiments with displayed and projected adversarial patches show that PercepGuard detects 96% of the attacks on average.

Our contributions are further summarized as follows.

- We propose PercepGuard that detects misclassification attacks against vision-based object detection and tracking (ODT) systems. PercepGuard is agnostic to attack methodologies, because we focus on the ODT output (e.g., object classes and trackers) instead of the input (image pixels).

- An RNN is used to classify the spatiotemporal fingerprint, inherent in a detected object's tracker, into an object class. We compare the result with ODT's classification result for attack detection. To improve the adversarial robustness, we adopt contextual information as additional features.

- We evaluate PercepGuard using a real-world dataset and a simulation dataset. It produces FPRs as low as 5% and TPRs as high as 99% against adaptive attackers. Real-world experiments where adversarial patches are realized using a TV and a projector yield a TPR of 96% on average.

- PercepGuard outperforms two baselines, NIC [46] and SCEME [10], in terms of the TP and FP trade-off. We demonstrate its extensibility by incorporating SCEME's context features. Sensitivity analysis also shows our RNN is not vulnerable to subtle, direct perturbation to its input.

1.2.4 Impact of Perception Attacks and Robust Prediction and Planning

Autonomous vehicles rely on perception to understand the environment. While a number of perception attacks have been proposed, their impact on the decision- making under various driving scenarios remains unclear. In this chapter, we take the first step and evaluate the outcome of object misclassification attacks, a representative category of perception attacks due to their consistently high attack success rates, where the attacker aims to alter the classification result of an obstacle via physically modifying the obstacle, or compromising the sensors. We develop a generic hijacking tool that could potentially allow us to evaluate a wide variety of perception attacks, individually or combined together at the same time, under different traffic scenarios in an end-to-end manner. Motivated by the findings from the hijacking tool, where an adversary can easily disrupt obstacle trajectory prediction hence the planning of autonomous vehicles using misclassification attacks, we propose a robust prediction system that remedies such class uncertainty.

The contributions of this chapter are summarized as follows.

- We are the first to evaluate at the system level the consequences of various perception attacks, individually or combined together, efficiently without attack implementation but propose to reproduce the attack outcome instead.
- We design and implement a general framework that allows us to hijack the communication among the decision-making modules, enabling arbitrary message modification.
- A robust object trajectory prediction system is proposed to incorporate classification uncertainty, where we adaptively choose the predictor based on the prior information and historical prediction performance.
- Evaluation results show that our framework is able to defeat adaptive misclassification, trajectory, and tracker hijacking attacks. Baseline comparison shows our system is more effective and efficient.

1.2.5 Intrusion Detection Evasion on In-Vehicle Networks

The Controller Area Network (CAN) is a bus standard commonly used in the automotive industry for connecting Electronic Control Units (ECUs) within a vehicle. The broadcast nature of this protocol, along with the lack of authentication or strong integrity guarantees for frames, allows for arbitrary data injection/modification and impersonation of the ECUs. While mitigation strategies have been proposed to counter these attacks, high implementation costs or violation of backward compatibility hinder their deployment. In Chapter 7, we focus on internal attacks where critical ECUs such as those that control the throttle and the brake are impersonated by one or more attacker-compromised ECUs. We reveal that existing CAN bus intrusion detection and identification systems are typically based on multiple frames to detect misbehavior and attribute it to a particular ECU; we show that such schemes are fundamentally vulnerable to hill-climbing-style attacks where the attacker carefully injects an increasing number of malicious messages which gradually shifts the fingerprints of the impersonated ECU to the attacker's. Motivated by the attack, a single-frame-based IDS system is developed and evaluated with real-world CAN bus voltage data collected by ourselves [47].

Our contributions are further summarized as follows.

- We demonstrate that a Hill-climbing-style attack can defeat *multi-frame* based intrusion detection and/or identification systems, in particular for vehicular CANs. We validate the effectiveness of our attack against two existing IDSs: physical layer-based Viden [23] and clock-based IDS CIDS [48].

- We collect CAN bus voltage data from real-world vehicles in motion, over the course of several months in order to track the physical characteristics under various operating conditions, such as temperatures. Thanks to this dataset, a *single-frame* based physical-layer identification solution SIMPLE can be developed that is fundamentally immune to hill-climbing attacks and can be evaluated under real-world data, which yields a low equal error rate (EER) around 0.8985%.

Chapter 2

Related Work

In this dissertation, we investigate the security of autonomous vehicles with machine learning models being used in the decision-making pipeline. Therefore, in this chapter we review related work on both machine learning security and autonomous system security.

2.1 Machine Learning Security

Despite the outstanding performance of deep learning models in image classification, object detection, etc. under non-adversarial scenarios, their adversarial robustness has been compromised by differing attacks. Here, we review the adversarial examples and the defenses against them.

2.1.1 Adversarial Examples

State-of-the-art adversarial examples can be categorized as digital or physical domain attacks. In digital domain attacks, the adversarial perturbation is directly applied to images (i.e., at the pixel level). Some of these are white-box attacks in that the adversary knows the architecture and the parameters of the targeted neural network [19, 20, 49–54]. Others are black-box attacks in which such parameters are unknown [55–59]. In physical domain attacks objects of interest are physically modified to cause misclassification [21, 52, 60–62], or to evade detection [63–65]. These physical domain attacks differ from our attacks in that we target the sensor (camera) without needing to physically modify any real-world object.

2.1.1.1 White-box Attacks

In a white-box attack, the adversary knows the architecture of the target network and the parameters of it. Under this scenario, adversarial examples were first discussed by Szegedy et al. [49], where they found an adversarial example of a benign image x by minimizing the L_2 norm of the perturbation δ, such that the classification result of $x + \delta$ was the targeted class t (an equality constraint), while the resulting image was still being valid (pixel values stayed within a certain range, i.e., a "box-constraint"). Carlini & Wagner [19] discussed multiple choices of the loss function and they eventually chose to use the difference of the logits value of the targeted class versus all the other classes. Moreover, Carlini & Wagner used the "change-of-variable" technique to address the box-constraint and finally they proposed an L_2 attack [19], which our approach will follow.

Since optimization-based methods above are sometimes slow, Goodfellow et al. proposed the Fast Gradient Sign Method (FGSM) [20] where they took the sign of the gradient of the loss function with respect to the input image for the directions of perturbation of pixels, along which the classification result will be most significantly altered. Similarly, Papernot et al. proposed JSMA [50] that crafted adversarial examples by computing forward derivatives to construct adversarial saliency maps for the directions. Moosavi et al. proposed DeepFool [51] that was based on an iterative linearization of the network. Although these methods are fast, they are not robust enough [39].

2.1.1.2 Black-box Attacks

In a black-box attack, the adversary does not know the architecture or the parameters of the network. However, the adversary can still generate adversarial examples based on local networks built up themselves and then apply these examples to target (remote) networks [56–58]. Nitin et al. avoided building a local network by querying the target network in order to estimate the gradient [55]. While the black-box attack model is more realistic, in this project we will focus on the white-box attack model as a first step, and extend to the black-box attack model in the future.

2.1.1.3 Physical Adversarial Examples

Since the adversary may not have direct access to the digital images in the real-world, researchers have proposed to implement the attack by altering physical objects to be imaged directly. Kurakin et al. printed out the digital adversarial examples generated using FGSM in an iterative manner [52], where they repeatedly called FGSM to update adversarial examples so that a higher attack success rate could be achieved. However, the printed examples could not stay misclassfied. Sharif et al. proposed an interesting method to generate physical adversarial examples in the shape of glasses [60]. An adversary who wore the glasses with special patterns would be able to fool a face recognition algorithm, impersonating another person. The adversary found the pattern δ by solving an optimization problem that considered both the smoothness and the printability of the patterns in addition to the attack robustness. Physical adversarial examples that targeted traffic signs were realized by Eykholt et al. [21], in which the adversary put stickers on a stop sign and made it misclassified as a yield sign. Similarly, the adversary found the pattern of stickers by considering the printability of the patterns, and also different angles, distance and lighting conditions, etc. The drawbacks of physical object modifications are that they may be noticeable to human observers and they may cause damage to the object permanently. Also, it may not be easy to gain physical access to the object in many cases.

2.1.2 Defenses against Adversarial Examples

In terms of defending neural networks from adversarial examples, be they physical or digital, schemes include modifying the network to be more robust [20, 39–41, 66–72], while other defenses have focused on either detecting adversarial inputs [46, 73–77] or transforming them into benign images [76, 78, 79], most of which are under the general assumption of bounded perturbations, hence are inapplicable to our attacks; while others could also be bypassed by being taken as constraints in the optimization formulation. As this work mainly focuses on sensor attacks, similar to [21, 60, 63, 80] we leave the validation of defenses as future work.

2.1.2.1 Network-oriented defenses

The earliest defense mechanisms adopted the idea of adversarial training [20, 40], where adversarial examples were used to train the network, but this method could not prevent future adversarial instances. Later, distillation training as a defense was proposed in [39] where soft-labels rather than hard-labels were used to train the network, but such defense was evaded by [19]. Inspired by cryptography, randomization-based defenses [66–68] were able to smooth the inter-class boundary but were also evaded by [81]. Certified defenses such as [69, 70] were proved to function only in small-scale networks. A recent certified defense [41] was able to scale but it only worked against norm-bounded attacks, i.e., the p-norm of the perturbation is bounded by a scalar, which some attacks, however, do not follow (e.g., spatial attacks [54]). Our attacks do not require to be norm-bounded either because there is not human in the loop to check whether or not an input image has been perturbed significantly.

2.1.2.2 Input-oriented defenses

Many works have been focusing on adversarial inputs and tried to transform them into benign images. For example, Meng et al. proposed MagNet [76], consisting of a detector network that learned to distinguish between normal and adversarial examples, and a reformer network that transformed adversarial examples into normal ones. Xu et al. quantized the input images in order to detect adversarial examples [73] based on the assumption that the adversarial perturbation was slight.

2.2 Autonomous Driving Security

Recent work has been focusing on the security of different modules in autonomous vehicles. Here, we review the state-of-the-art attacks and defenses in these fields.

2.2.1 Perception Module Security

Perception the first component of the decision-making pipeline and the most exposed one, which creates a large attack surface. In this section we review the perception attacks and countermeasures.

2.2.1.1 Perception Attacks

Although the perception systems have shown promising results in non-adversarial scenarios, they are however vulnerable to various perception attacks. There are creation attacks [6,9,63] where non-existing objects are forged, and removal attacks [8, 9, 63, 82, 83] where existing objects are erased, and misclassification attacks [6, 8, 21, 22], where the attacker alters the video so that the algorithm classifies the victim object as an attack-specified class. The alteration may occur in either of the three domains:

Physical Domain An attacker can place a sticker on a stop sign which will then be misclassified as a yield sign [21], or mount a TV on the back of the preceding vehicle so that it is recognized as a person [4], or simply project a person on the ground or wall [22], etc.

Perception Domain An attacker may shine light into the camera to create adversarial ghosts [6,7], use acoustic wave to disrupt the stabilizer of the camera [9], or exploit the rolling shutter effect [8] etc. Existing remote attacks against cameras [25–27] are denial-of-service attacks and do not seek to compromise the object classifier as our attacks do. Those attacks that do target object classification [21, 63, 84] are either digital or physical domain attacks (i.e., they need to modify the object of interest in this case a traffic sign or road pavement, physically or after the object has been captured by a camera) rather than perception domain attacks [29,30]. Li et al. [85]'s attacks on cameras require attackers to place stickers on lenses, to which is generally hard to get access. Similarly, several light-based attacks [61, 86, 87] fall within the domain of physical attacks, as opposed to our perception domain attack, because these approaches illuminate the object of interest with visible or infrared light. We did not consider infrared noise in our attacks as it can be easily eliminated from visible light systems using infrared filters.

Attacks on LiDAR systems [25, 80, 88, 89] are first discussed in [25, 88], though these works do not consider a neural network backend for object classification. Two other works [80, 89] do describe a variant of a LiDAR attack meant to subvert an associated object detection system, but they are considerably easier to carry out than our visible light-based attacks against cameras because attackers can directly inject adversarial laser pulses into LiDARs

without worrying about blocking the object of interest.

Digital Domain This is the strongest threat as the attacker have access to the sensing output (e.g., image) and can arbitrarily modify an image/video at the pixel level [90]. Traditional digital domain adversarial examples [19] assume that the attacker induced noise is norm-bounded, to avoid being detected by human users. However, in the context of autonomous system applications, this constraint is no longer relevant since the images/videos are not examined by human users. Thus, we do not consider digital domain attacks in this chapter, because if the attacker can gain direct access to the sensing output, they can also have the ability to directly modify, for instance, the outcome of perception module/decisions, and detecting such attacks falls into the goal of intrusion detection systems [44].

2.2.1.2 Countermeasures

Traditional defenses against adversarial examples [39, 40, 76, 91] are not applicable to perception attacks as they have a general assumption of norm-bounded perturbation which cannot apply to physical/perceptional attacks where typically the noise are not bounded by a small norm, otherwise they would be nearly impossible to realize. Defenses that do not have such an assumption limit themselves to a particular attack methodology, e.g., adversarial patch attacks [35, 92] or physically-realizable attacks [93]. Because PercepGuard verifies only the output of ODT systems (Fig. 5.2), it is agnostic to different attack methodologies and object detection algorithms.

Here, we focus more on consistency-based defenses for vision systems; they can be categorized into two types: sensor fusion-based and single sensor-based. Consistency check can be done across multiple sensors, be they heterogeneous [33, 36] or homogeneous [37]. For example, model-based methods such as Savior [33] verify physical invariants across multiple, heterogeneous sensors which requires sophisticated physical models, while we take a data-driven approach that is easier to deal with uncertainty and consider additional contexts. Zhang et al. [37] propose to fuse different views from multiple cameras to detect DoS attacks on cameras [25] but they are vulnerable to non-DoS camera attacks [6, 7].

On the other hand, one can use a single sensor's output, but with empirical/semantic

knowledge that verifies the integrity of the sensory data. For example, AdvIT [34] detects adversarial noise from videos based on optical flow consistency, however it is limited to norm-bounded perturbation, thus vulnerable to universal adversarial patches [94] which are able to maintain the flow consistency across multiple frames. Co-existence consistency among multiple objects can also be utilized (e.g., traffic signs more likely co-exist with cars than a sink), thus incoherent objects can be detected [10, 95], although these methods do require multi-object scenes and may subject to the richness of context and adaptive attacks [96].

For a single object, recently Gurel et al. proposed a knowledge-enhanced ML pipeline that verifies the consistency of a static object's (e.g., a traffic sign) attributes, such as color, text, and shape [38, 97], while Wang et al. apply a similar idea to person re-identification [98]. However, their methodology is inapplicable to modeling the spatiotemporal consistency of moving objects, because internal attributes for static objects are discrete and finite, which can be specified by domain experts. However, spatiotemporal features are much more complex as they are continuous and high-dimensional, and involves more uncertainty. For example, having an octagon shape is a necessary condition for a STOP sign [38], but the movement of a vehicle cannot be described in such a deterministic manner.

Moving target defenses were proposed against traditional adversarial examples [19] where slightly different machine learning models are randomly selected from a model pool for deployment which makes the attack optimization problems more difficult to solve [99, 100], however these methods do assume norm-bounded noise. Occluding relations among objects can be used to detect LiDAR attacks [31, 80], but such approaches are only applicable to LiDAR-based systems.

The idea of using temporal consistency and multimodal classifiers has been applied to other domains such as biometrics-based user authentication/identification [101–107] and human activity/emotion recognition [108–113] etc. For example, to authenticate users, one can learn the time-series data from multimodal sensors that are commonly available on a mobile phone, such as the pressure sensor when they are entering their PIN [106], or the accelerometer and the gyroscope [103] whose data can also be learned using an RNN [101]. Wearables such as smartwatches that are equipped with similar sensors can also help authenticate users [102], e.g. by their keystroke dynamics [105]. Driver stress detection can be done by fusing eye

movement, ego-vehicle dynamics, and the environmental contexts utilizing the CNN-LSTM models [109]. Blood flow from retina scanning can be used for liveness detection [107]. However, it is infeasible for us to directly apply these methods for autonomous driving since our goal is to detect object misclassification in perception with the consideration of adaptive attacks. Moreover, due to the different types of features we need to design our own system architecture.

2.2.2 CAN Bus Intrusion Detection

The CAN protocol was introduced in 1986 by Robert Bosch GmbH [114] for in-vehicle communication. Due to the lack of authentication, threats have been demonstrated with an attacker who gains access to the CAN bus and launch denial of service, or impersonation attacks [24, 42, 43, 115, 116].

Physical layer identification (PLI) [117] has been proposed to utilize the hardware manufacturing inconsistency to fingerprint ECUs for intrusion detection. CAN Bus IDS can be categorized into timing-based and voltage-based.

2.2.2.1 Timing-based

CAN messages are typically sent in a periodic manner. The analysis on the interval/frequency of the messages might be able to show the first evidence of intrusion. Müter et al. [118] use entropy to analyze the randomness of intervals. Moore et al. [119] proposed a data-driven inter-signal arrival times model to detect injection attacks. Abnormality detection sensors [120] evaluate the payload length, frequency or correlation, etc. Taylor et al. [121] proposed to detect anomalies of the sequence data transmitted from an ECU by applying neural networks. A similar work [122] automatically classifies the fields in CAN messages and measures valid ranges based on previous data, which is computationally expensive. Ying et al. proposed [123] TACAN based on shared cryptographic keys and inter-arrival times, which introduced computational overhead. Cho and Shin [24] proposed clock skew based fingerprints for their PLI-based IDS, CIDS. However, the timing-based approaches can be easily defeated by an adversary imitating the target ECU's timing behavior [48, 124]. Avatefipour et al. [125] train a neural network to capture the features in both frequency and time domain, which

is impractical for a CAN bus because of an ECU's weak computational ability. Besides, timing-based approaches naturally require multiple frames to make detection/identification decisions, which also makes them vulnerable to the Hill-climbing attack. Dagan et al. [126] introduce a software patch as an anti-spoofing tool to disable the compromised ECU.

2.2.2.2 Voltage-based

The voltage output nature differs among the transmitters of ECUs, enabling voltage-based approaches to detect the intrusion and identify its source. Murvay et al. [127] took the first step towards voltage identification, which is later extended by several contributions such as Viden [23], VoltageIDS [48], and Scission [44]. These methods construct and update voltage profiles for the ECUs and use them for identification of malicious messages. However, the effect of the variations of voltage power source and ambient temperature on these features is non-negligible and has not been taken into consideration. That is, the life-span of the valid features would be limited to a short interval as the source voltage would be easily affected by any change in ambient temperature. This is the reason the features need to be updated every time the vehicle is restarted, which motivates applying adaptive profile updates [23]. In fact, because the features smoothly evolve through time, Viden has to keep track of these changes by updating their profiles through time, which makes Viden vulnerable due to the lack of secure training data that updates with time.

Chapter 3

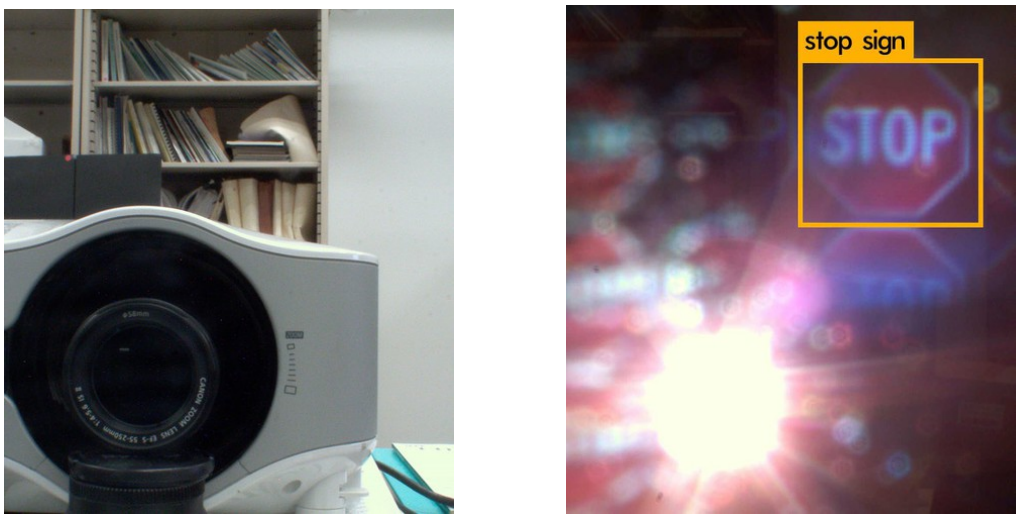
Remote Perception Attacks against Camera-based Object Recognition Systems

3.1 Introduction

Object detection and classification have been widely adopted in autonomous systems, such as automated vehicles [14, 17] and unmanned aerial vehicles [128], as well as surveillance systems, e.g., smart home monitoring systems [129, 130]. These systems first perceive the surrounding environment via sensors (e.g., cameras, LiDARs, and motion sensors) that convert analog signals into digital data, then try to understand the environment using object detectors and classifiers (e.g., recognizing traffic signs or unauthorized persons), and finally make a decision on how to interact with the environment (e.g., a vehicle may decelerate or a surveillance system raises an alarm).

While the cyber (digital) attack surface of such systems have been widely studied [43, 131], vulnerabilities in the perception domain are less well-known, despite perception being the first and critical step in the decision-making pipeline. That is, if sensors can be compromised then false data can be injected and the decision making process will indubitably be harmed as the system is not acting on an accurate view of its environment. Recent work has demonstrated false data injection against sensors in a remote manner via either electromagnetic (radio frequency) interference [132], laser pulses (against microphones [133], or LiDARs [25, 80, 88]), and acoustic waves [134, 135]. These perception domain sensor attacks alter the data at the source, hence bypassing traditional digital defenses (such as crypto-based authentication or access control), and are subsequently much harder to defend against [29, 30]. These attacks can also be remote in that the attacker needn't physically contact/access/modify devices or objects.

Recently, several sensor attacks against autonomous vehicles have been proposed, including



(a) Projector off **(b)** Projector on

Figure 3.1: A STOP sign image was injected into a camera by a projector, which was detected by YOLOv3 [1].

camera attacks [25, 26], and LiDAR attacks . Different from vision-based imaging systems, LiDARs are mainly used for obstacle detection and ranging. LiDAR attacks are relatively easier to carry out because in order to create a fake object, the attacker could directly inject laser pulses. Also, LiDAR is not as widely adopted as cameras. For example, Tesla cars are not equipped with LiDARs [14]. Camera-based image classification systems, on the other hand, they are generally harder to spoof because (unlike LiDAR attacks) the attacker cannot block the target object (otherwise it becomes trivial), and has to carefully control the injection intensity so as not to blind the camera. Limited existing camera attacks' goal was to blind the cameras (denial-of-service), instead of spoofing object detection or classification.

Among the aforementioned sensors, at least for automated systems in the transportation and surveillance domains, cameras are more common/crucial. Existing *remote* attacks against cameras are limited to, essentially, denial-of-service attacks [25–27], which are easily detectable (e.g., by tampering detection [28]) and for which effective mitigation strategies exist (e.g., by sensor fusion [136]). In this work, we consider attacks that cause camera-based image classification system to either misperceive actual objects or perceive non-existent objects by remotely injecting light-based interference into a camera, without blinding it. Formally, we consider *creation attacks* whereby a spurious object (e.g., a non-existent traffic sign, or obstacle) is seen to exist in the environment by a camera, and *alteration attacks*, in which an

existing object in the camera view is changed into another attacker-determined object (e.g., changing a STOP sign to a YIELD sign or changing an intruder into a bicycle).

As it is not possible, due to optical principles, to directly project an image into a camera, we propose to exploit two common effects in optical imaging systems, viz., *lens flare effects* and *exposure control* to induce camera-based misperception. The former effect is due to the imperfection of lenses, which causes light beams to be refracted and reflected multiple times resulting in polygon-shape artifacts (a.k.a., *ghosts*) to appear in images [137]. Since ghosts and their light sources typically appear at different locations, an attacker can overlap specially crafted ghosts with the target object's without having the light source blocking it. Auto exposure control is a feature common to cameras that determines the amount of light incident on the imager and is used, for example, to make images look more natural. An attacker can leverage exposure control to make the background of an image darker and the ghosts brighter, so as to make the ghosts more prominent (i.e., noticeable to the detector/classifier) and thus increase attack success rates. Fig. 3.1 presents an example of a creation attack, where we used a projector to inject an image of a STOP sign in a ghost, which is detected and classified as a STOP sign by YOLOv3 [1], a state-of-the-art object detector.

Theoretically arbitrary patterns can be injected via ghosts. However, it is challenging to practically and precisely control the ghosts, in terms of their resolutions and positions in images, making arbitrary injection impracticable in some scenarios. Hence, we propose an empirical projector-camera channel model that predicts the resolution and color of injected ghost patterns, as well as the location of ghosts, for a given projector-camera arrangement. Experimental results show that at short distances attack success rates are as high as 100%, but at longer distances the rates decrease sharply; this is because at long distances ghost resolutions are low, resulting in patterns that cannot be recognized by the classifier.

To improve the efficacy of our attack, which we dub GhostImage, especially at lower resolutions, we assume that the attacker possesses knowledge about the image classification/detection algorithm. Based on this knowledge the attacker is able to formulate and solve an optimization problem to find optimal attack patterns, of varying resolutions, to project that will be recognized by the image classifier as the intended target class [19, 49]; i.e., the pattern projected will yield a classification result of the attacker's choice. As the channel

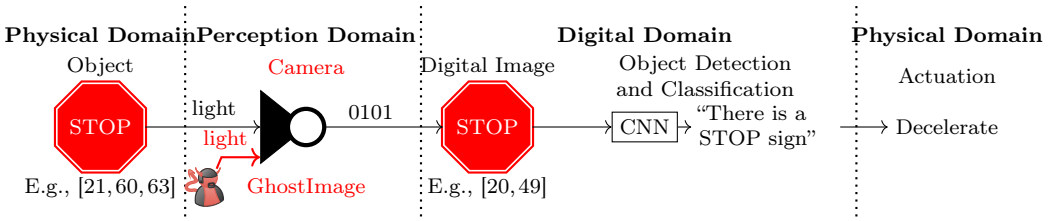


Figure 3.2: Camera-based object classification systems. GhostImage attacks target the perception domain, i.e., the camera.

may distort the injected image (in terms of color, brightness, and noise), we extend our projector-camera model to include auto exposure control and color calibration and integrate the channel model into our optimization formulation. This results in a pattern generation approach that is resistant to channel effects and thus able to defeat a classifier under realistic conditions.

We use self-driving and surveillance systems as two illustrative examples to demonstrate the potential impact of GhostImage attacks. Proof-of-concept experiments were conducted with different cameras, image datasets, and environmental conditions. Results show that our attacks are able to achieve attack success rates as high as 100%, depending on the projector-camera distance.

The rest of the chapter is organized as follows. We will setup our system model, attack model and threat model in Section 3.2, and introduce some preliminaries in Section 3.3. In Section 3.4 we will present our naive GhostImage attacks which covers how we can project arbitrary patterns in the image, and then in Section 3.5 we will discuss advanced GhostImage attacks by which we can find better patterns to project. We extensively evaluate our attacks in Section 3.7, and discuss about the attack in Section 3.8. We conclude our work in Section 3.9.

3.2 System and Threat Model

System and attack models are described, including two attack objectives and the attacker's capabilities.

3.2.1 System Model

We assume an end-to-end camera-based object classification system (Fig. 3.2) in which a camera captures an image of a scene with objects of interest. The image is then fed to an object detector to crop out the areas of objects, and finally these areas are given to a neural network to classify the objects. Autonomous systems increasingly rely on such classification systems to make decisions and actions. If the classification result is incorrect (e.g., modified by an adversary), wrong actions could be taken. For example, in a surveillance system, if an intruder is not detected, the house may be broken-in without raising an alarm.

While there are attacks focusing on the physical domain [52, 60], where the attackers physically change the object, and other attacks working in the digital domain, trying to perturb digital images [19, 50], our GhostImage attacks target the perception domain, where sensing mechanisms are compromised. Most of existing sensor attacks focused on the sensors alone, while AdvLiDAR [80] attack LiDAR sensors as a perception step in an end-to-end autonomous system. Rather than attack LiDAR sensors, in this work we attack cameras, a kind of sensors that are much more widely used on autonomous vehicles, as well as surveillance systems, etc.

3.2.2 Threat Model

We consider two different attack objectives. In **creation attacks** the goal is to inject a spurious (i.e., non-existent) object into the scene and have it be recognized (classified) as though it were physically present. For **alteration attacks** an attacker injects adversarial patterns over an object of interest in the scene that causes the object to be misclassified.

There are two types of attackers with differing capabilities: **Camera-aware attackers** who possess knowledge of the victim's camera (i.e., they do not know the configuration of the lens system, nor post-processing algorithms, but they can possess the same type of camera used in the target system), from which they can train a channel model using the camera as a black-box. With such capabilities, they are able to achieve creation attacks and alteration attacks. **System-aware attackers** not only possess the capabilities of the camera-aware attackers, but also know about the image classifier including its architecture and parameters,

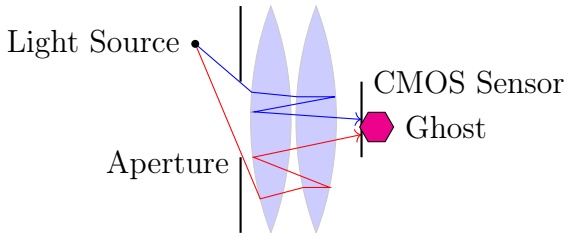


Figure 3.3: Ghost effect principle

i.e., black-box attacks on the camera but white-box attacks on the classifier. With such capabilities, it is able to achieve creation attacks and alteration attacks as well, but with higher attack success rates.

Both types of attackers are remote (unlike the lens sticker attack [85]), i.e., they do not have access to the hardware or the firmware of the victim camera, nor to the images that the camera captures. We assume that both attackers are able to track and aim victim cameras [27, 80, 138].

3.3 Background

In this section, we will introduce optical imaging principles, including flare/ghost effects and exposure control, which we will exploit to *realize* GhostImage attacks. Then, we will discuss the preliminaries about neural networks and adversarial examples that we will use to *enhance* GhostImage attacks.

3.3.1 Optical Imaging Principles

Due to the optical principles of camera-based imaging systems, it is not feasible to directly point a projector at a camera, hoping that the projected patterns can appear at the same location with the image of the targeted object, because the projector has to obscure the object in order to make the two images overlap. Instead, we exploit lens flare effects and auto exposure control to inject adversarial patterns.

Lens flare effects [137,139] refer to a phenomenon where one or more undesirable artifacts appear on an image because bright light get scattered or flared in a non-ideal lens system (Fig. 3.3). Ideally, all light beams should pass directly through the lens and reach the CMOS

sensor. However, due to the quality of the lens elements, a small portion of light gets reflected several times within the lens system and then reaches the sensor, forming multiple polygons (called "ghosts") on the image. The shape of polygons depends on the shape of the aperture. For example, if the aperture has six sides, there will be hexagon-shaped ghosts in the image. Normally ghosts are very weak and one cannot see them, but when a strong light source (such as the sun, a light bulb, a laser, or a projector) is present (unnecessarily captured by the CMOS sensor), the ghost effects become visible. Fig. 3.3 shows only one reflection path, but there are many other paths and that is why there are usually multiple ghosts in an image.

Existing literature [137] about ghosts focused on the simulation of ghosts given the detailed lens configurations, in which the algorithms simulate every possible reflection path. Such white-box models are computationally expensive, and also requires white-box knowledge of internal lens configurations, thus are not suitable for our purposes. In Sections 3.4 and 3.5, we study flare effects in a black-box manner (more general than Vitoria et al. [139]), where we train a lightweight end-to-end model that is able to predict the locations of ghosts, estimate the resolutions within ghost areas, and also calibrate colors.

Exposure control mechanisms [140] are often equipped in cameras to adjust brightness by changing the size of the aperture or the exposure time. In this work, we will model and exploit auto exposure control to manipulate the brightness balance between the targeted object and the injected attack patterns in ghosts.

3.3.2 Neural Nets and Adversarial Examples

We abstract a neural network as a function $Y = f_\theta(x)$. The input $x \in \mathbb{R}^{w \times h \times 3}$ (width, height and RGB channels) is an image, $Y \in \mathbb{R}^m$ is the output vector, and θ is the parameters of the network (which is fixed thus we omit it for convenience). A softmax layer is usually added to the end of a neural network to make sure that $\sum_{i=1}^{m} Y_i = 1$ and $Y_i \in [0, 1]$. The classification result is $C(x) = \text{argmax}_i Y_i$. Also, the inputs to the softmax layer are called *logits* and denoted as $Z(x)$.

An adversarial example [49] is denoted as y, where $y = x + \Delta$. Here, Δ is additive noise that has the same dimensionality with x. Given a benign image x and a target label t, an adversary wants to find a Δ such that $C(x + \Delta) = t$, i.e., *targeted attacks*. Note that, in this

chapter, the magnitude of Δ is not constrained below a small threshold, since the perceived images are usually not directly observed by human users. But we still try to minimize it because it represents the attack power and cost.

3.3.3 Sensor Attacks

Perception in autonomous and surveillance systems occurs through sensors, which convert analog signals into digital ones that are further analyzed by computing systems. Recent work has demonstrated that the sensing mechanism itself is vulnerable to attack and that such attacks may be used to bypass digital protections [29, 30]. For example, anti-lock braking system (ABS) sensors have been manipulated via magnetic fields by Shoukry et al. [141], microphones have been subject to inaudible voice and light-based attacks [133, 142], and light sensors can be influenced via electromagnetic interference to report lighter or darker conditions [132]. The reader is referred to [29, 30] for a review of analog sensor attacks.

3.4 Camera-aware GhostImage Attacks

In this section, we will discuss how a camera-aware attacker is able to inject arbitrary patterns in the perceived image of the victim camera using projectors. While there could be many options (e.g., lasers, bulbs, flashlights, etc.) for the source of the light noise, in this work we simply use projectors, because they can inject different colors in different shapes while the others cannot.

3.4.1 Technical Challenges

Since we assume that the attacker do not have access to the images that the targeted camera captures, they will have to be able to predict how ghosts might appear in the image. First, the locations of ghosts should be predicted given the relevant positions of the projector and the camera, so that the attacker can align the ghost with the image of the object of interest to achieve alteration attacks. Second, since a projector can inject shapes in ghost areas, the attacker needs to find out the maximum resolution of shapes that it can inject. Lastly, it is also challenging to realize the attacks derived from the position and resolution

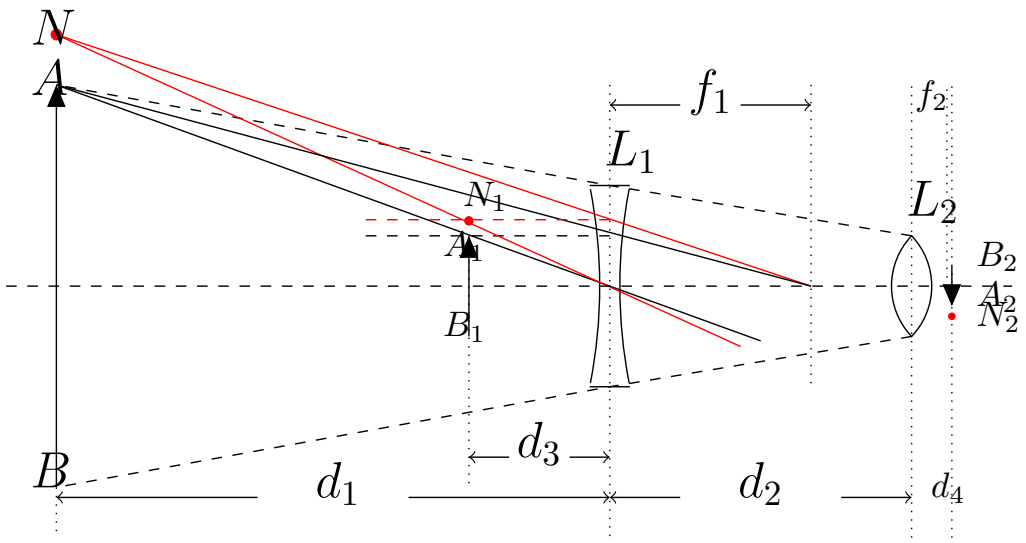


Figure 3.4: Noise N cannot overlap with the image of the object AB even with an additional concave lens.

models above with a limited budget.

3.4.1.1 Direct projection using an additional lens

As a motivation why we need to leverage ghost effects rather than directly inject light beams, here we present why the latter is not feasible.

Lemma 3.4.1. *However the additional lenses are placed, the image of noise cannot overlay with the image of the object without obscuring the object.*

Proof. We are going to prove that even with an additional concave lens. See Figure 3.4 for a diagram, where a concave lens L_1 is placed between an object AB and a camera's convex lens L_2. L_1's focal length is f_1 and L_2's is f_2. The distance between L_1 and L_2 is d_2. The noise source N is right upon A. From the perspective of L_2, AB is completely obfuscated by L_1; in other words, all light rays of AB that go through L_2 must at first go through L_1. Both the object and the noise source share the same object distance to L_1, which is d_1. The image of AB formed by L_1 is A_1B_1, and N_1 is the image of N. The distance between A_1B_1 and L_1 is d_3. The image of formed by L_2 is A_2B_2 and N_2. The distance between A_2B_2 and L_2 is d_4.

In order to solve a problem consisting of multiple lenses, we usually analyze each lens individually and sequentially. That is, we calculate the image formed by the first lens, then

use that image as the input to the second lens.

For L_1, the input is AB, thus we have

$$\frac{1}{d_1} + \frac{1}{d_3} = \frac{1}{f_1}, \tag{3.1}$$

and the magnification is calculated as

$$M = \frac{|N_1B_1|}{|NB|} = \frac{|A_1B_1|}{|AB|} = \frac{|A_1N_1|}{|AN|} = \frac{d_3}{d_1}, \tag{3.2}$$

from which we know that N_1 does not overlap with A_1B_1.

For L_2, the input is A_1B_1 (L_2 cannot "see" AB directly because AB is completely obfuscated by L_1), thus we get

$$\frac{1}{d_2 + d_3} + \frac{1}{d_4} = \frac{1}{f_2}, \tag{3.3}$$

and the magnification is calculated as

$$M = \frac{|N_2B_2|}{|N_1B_1|} = \frac{|A_2B_2|}{|A_1B_1|} = \frac{|A_2N_2|}{|A_1N_1|} = \frac{d_4}{d_2 + d_3}, \tag{3.4}$$

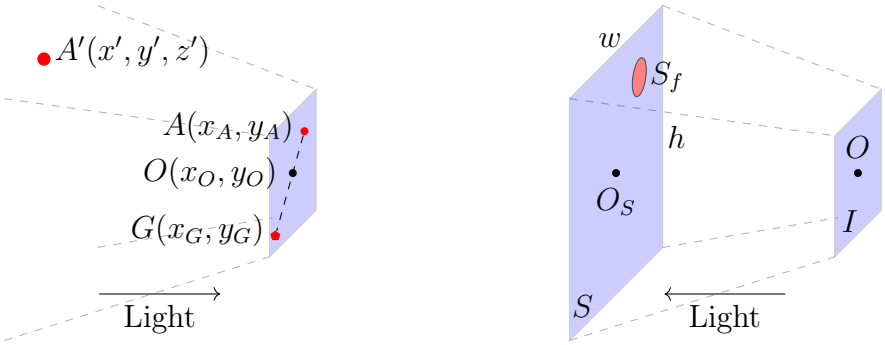
from which we know that N_2 does not overlap with A_2B_2. As a result, no matter how we place the additional concave lens, we cannot apply the noise to the image of the object without obscuring the object. The proof for a convex lens follows the same logic.

□

3.4.2 Ghost Pixel Coordinates

Given the pixel coordinates of the target object G (Fig. 3.5a), we need to derive the real-world coordinates A' of the projector so that we know where to place the projector in order to let one of the ghosts overlap with the image of the object. To do this, we derive the relationship between G and A' in two steps: We first calculate the pixel coordinates of the light source A given A', and then we calculate G based on A.

Based on homogeneous coordinates [143], assuming the camera is at the origin of the



(a) Capture **(b)** Projection

Figure 3.5: Capture and projection are reverses of each other.

coordinate system, we have

$$(u, v, w)^\top = M_c \cdot (x', y', z', 1)^\top, \tag{3.5}$$

where M_c is the camera's geometric model [143], a 3×4 matrix. M_c can be trained from another (similar) camera, and then be applied to the victim camera. The coordinates of A is then $A = (x_A, y_A)^\top = (u/w, v/w)^\top$, by the homogeneous transformation. Note that, A does not have to appear in the view of the camera, which makes the attack stealthier [144].

In order to find the relationship of the pixel coordinates between light sources A and their ghosts G, we did a simple experiment where we moved around a flashlight in front of the camera [145], and recorded the pixel coordinates of the flashlight and the ghosts. Similar to Vitoria et al.'s results [139], we observe that, for each G, we have $\overline{AO_I}/\overline{O_IG} = r_G$ (being

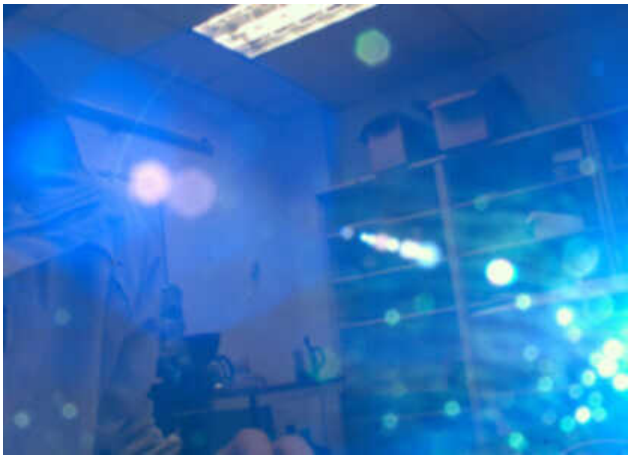


Figure 3.6: There are ghosts but the light source is out of view.

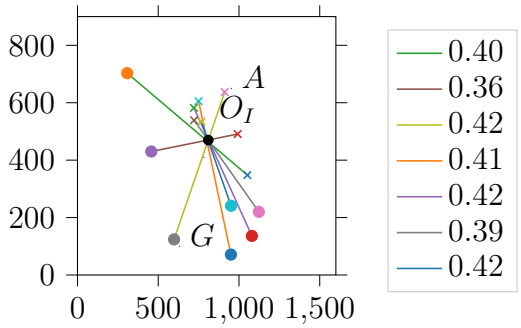


Figure 3.7: Ghost position v.s. light source position. Crosses are light source at different locations and the circles are the according biggest ghosts (as examples).

constant), wherever A is, and $r \in (-\infty, \infty)$. This means the feasible region for the placement of the projector is large; to attack an autonomous vehicle, for example, it can be located on an overbridge, on a traffic island, or even in the preceding vehicle or on a drone, etc. Finally, given $A = (x_A, y_A)$, $O_I = (x_O, y_O)$ and r, we can derive the coordinates of ghosts as

$$G = \begin{pmatrix} x_O - (x_A - x_O)/r \\ y_O - (y_A - y_O)/r \end{pmatrix}. \tag{3.6}$$

With G's coordinates, the attacker is able to predict the pixel location of ghosts and try adjusting the position and orientation (which implies the angle) of the light source in the real world so as to align one or more ghosts with the image of the object, whose pixel coordinates can be derived using (3.5), too. Note that the light source does not need to be within the field of view (Fig. 3.6).

3.4.3 Ghost Resolution

In our daily life, ghosts normally appear as pieces of single-color polygon-shaped artifacts; this is because the light sources that cause these regular ghosts are single-point sources of light that have just one single color, such as light bulbs, flashlights, etc. In this work, however, we find out that one is able to bring patterns into these ghost areas by simply using a low-cost projector, a special source of light that shines variant patterns in variant colors. For example, in Fig. 3.1, an image of a STOP sign that is projected by a projector, appears in one of the ghost areas in the image; this is because the pixel resolution of the projector is high enough

that multiple light beams in different colors (got reflected among lenses and then) go into the same ghost. In this subsection, we study the resolution of the patterns in ghost areas [1].

Let us first define the *throwing ratio* of a projector. In Fig. 3.5b, let plane S be the projected screen (e.g. on a wall), whose height and width are denoted as h and w, respectively. The distance $d = \overline{O_S O_I}$ is called the throwing distance. The throwing ratio of this projection is $r_{\text{throw}} = d/w$. The (physical) size of the projected screen at the victim camera's location is denoted S_O, a part of which is captured by the CMOS sensor of the camera in the ghost area, and we denote the (physical) size of that area as S_f. Let us also define the resolution of the entire projected screen as P_O in terms of pixels (e.g., 1024×768), and the resolution of the ghost as P_f. Clearly, there is a linear relationship among them: $P_f/P_O = S_f/S_O$, where $S_O = wh$. Finally, we can calculate the resolution of the ghost given d and r_{throw}:

$$P_f = \frac{P_O S_f}{\frac{h}{w}\left(\frac{d}{r_{\text{throw}}}\right)^2}. \tag{3.7}$$

Here, S_f is a constant because the size of the lens is fixed; e.g., the camera [145] has $S_f = 0.0156$ cm^2. We can obtain S_f by deriving from (3.7) only once for one camera. S_f of our camera can be calculated once and for all, as

$$S_f = \frac{P_f S_O}{P_O} = \frac{32 \times 32 \cdot 4\text{cm} \times 3\text{cm}}{1024 \times 768} = 0.0156 \text{ cm}^2,$$

because we did an experiment where we observed $P_f = 32 \times 32$ resolution in the ghost, when the projected screen size was $S_P = 4 \times 3$ cm^2, and the resolution of the projector was set as $P_O = 1024 \times 768$.

3.4.4 Attack Realization and Experiment Setup

According to Eq. 3.7, if the attacker wants to carry out long-distance and high-resolution GhostImage attacks, he/she needs a projector with a large throwing ratio r_{throw}. However, the factory longest-throw lenses (NEC NP05ZL Zoom Lens [146]) of our projector can achieve

[1] We are interested in the resolution of the projector pixels, not camera pixels; a projector pixel is usually captured by multiple camera pixels.

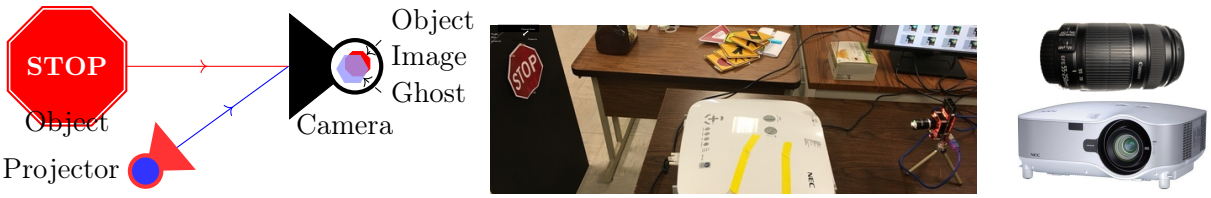


Figure 3.8: (Left) Attack setup diagram. (Middle) In-lab experiment setup. (Right) Attack equipments: We replaced the original lens of the NEC NP3150 Projector [2] with a Canon EFS 55-250 mm zoom lens [3].

a throwing ratio of maximum 7.2 (which means 9×9 at one meter), and expensive (about $1600). Instead, we use a cheap ($80) zoom lens (Fig. 3.8, Right) [3] that was originally designed for Canon cameras. In our experiments, such a configuration is interestingly feasible[2] (Fig. 3.8), achieving the maximum throwing ratio of 20 when the focal length is 250 mm, which means that at a distance of one meter, 32×32-resolution attacks can be achieved. See Sec. 3.8.1 for more discussion on lens and projector selection. Note that there are other telephoto zoom lenses with much longer focal lengths, e.g., for $200, one can acquire an Opteka telephoto zoom lens [147] that supports up to 1300 mm, which means a throwing ratio of roughly 100 and the max resolution of 160×160 at one meter.

Fig. 3.8 (left) shows a general diagram of GhostImage attacks, where the light source (i.e., a projector) is pointing at the camera from the side, so that the camera can still capture the object (e.g., a STOP sign) for alteration attacks. The light source injects light interference (marked in blue) into the camera, which gets reflected among the lenses of the camera, resulting in ghosts that overlap with the object in the image. Accordingly, a photo of our in-lab experiment setup is given in Fig. 3.8. The Canon lens was loaded in the NEC projector, though it cannot be seen in the photo. We will evaluate our attack on three different cameras (Sec. 3.7.2.3). We use an Aptina MT9M034 camera (designed for autonomous driving) [145] and a Ring indoor security camera (for surveillance systems) [148] as the targeted cameras (Section 3.7.2.3).

To mount a creation attack, the attacker computes the maximum resolution P_f for the ghost with a distance d based on (3.7), and then *downsamples* the target image to the resolution P_f The reason for downsampling can be seen from (3.7) that, with a fixed throwing

[2]Because projectors and cameras are dual devices (Fig. 3.5), their lenses are interchangeable.

ratio r_{throw}, larger distances d result in lower resolutions P_f. Hence, when the projector is far away from the camera, the attacker is not able to inject the entire attack image (in its original high resolution) in a ghost area. As a solution, the attacker downsamples the attack images in order to fit in the ghost area. The attacker chooses downsampling as a heuristic approach because he/she is not aware of the classification algorithm.

To mount alteration attacks, in addition to (3.7) for downsampling, the attacker also needs to consider the pixel coordinates (Eq. 3.6) of the ghost because the attacker needs to align the ghost with the image of the object of interest so that the resulting, combined image deceives the classifier.

3.4.5 Camera-aware Attack Evaluation

We substantiate camera-aware attacks on an image classification system that we envision would be used for automated vehicles. Specifically, images, taken by an Aptina MT9M034 camera [145], are fed to a traffic sign image classifier trained on the LISA dataset [149]. In Sec. 3.7, we will evaluate classification systems for other applications, with different cameras and different datasets.

3.4.5.1 Dataset and neural network architecture

In order to train an unbiased classifier, we selected eight traffic signs (with 80 instances) from the LISA dataset [149] (Fig. 3.19a). The network architecture is identical to [21]. We used 80% of samples from the balanced dataset to train the network and the rest 20% to test the network; it achieved an accuracy of 96%.

3.4.5.2 Evaluation methodology

We iterated five distances, m source classes, m target classes. For each target class, we sampled k images randomly from the dataset. For every combination, we first downsampled the target image based on (3.7), and projected the image at the camera using the NEC projector. We then took the captured image, cropped out the ghost area, and used the classifier to classify it. If the classification result is the target class, we count it as a successful attack. The procedure for creation attacks is slightly different: Rather than printed traffic

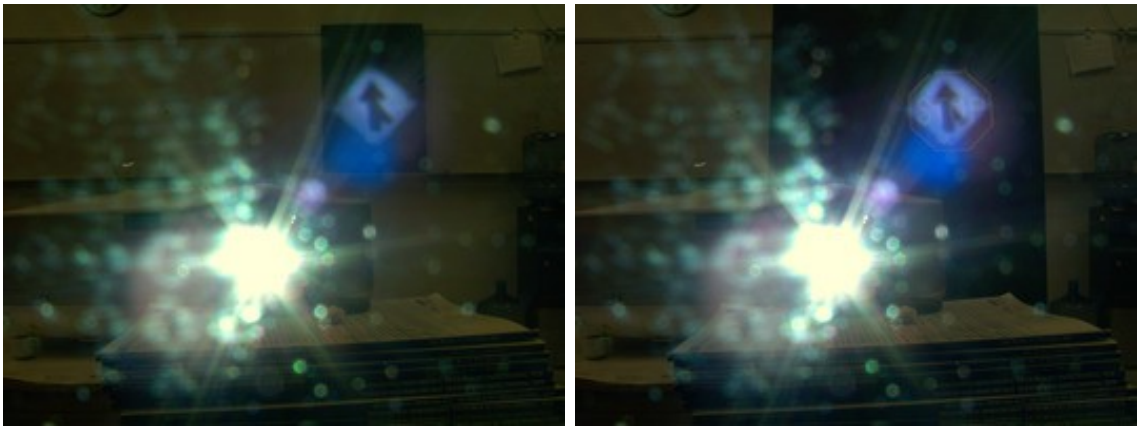


Figure 3.9: Camera-aware attack examples at one meter in perception domain. Left: Creating a Merge sign. Right: Altering a STOP sign (in the background) into a Merge sign.

signs, we placed a blackboard as the background as it helped us locate the ghosts. Given a throwing radio of 20 (thanks to the Canon lens) we evaluated five different distances from one meter to five meters. Based on (3.7), they resulted in 32×32, 16×16, 8×8, 4×4, and 2×2 resolutions, respectively.

3.4.5.3 Results

The results about attack success rates of camera-aware attacks at varying distances are shown in Table 3.1 (Fig. 3.9 illustrates two successful camera-aware attacks). For the digital domain, we simply added attack images Δ on benign images x as $y = (x + \Delta)/\|x + \Delta\|_\infty$. Based on these experiments, we observe: First, as the distance increases, the success rate decreases. This is because lower-resolution images are less well recognized by the classifier. Second, digital domain results are better than perception domain one, because images are

Table 3.1: Camera-aware attack success rates

Distances	Creation Attacks		Alteration Attacks	
(meter)	Digital	Perception	Digital	Perception
1	98%	41%	95%	33%
2	98%	36%	88%	33%
3	80%	34%	67%	34%
4	36%	15%	28%	10%
5	14%	10%	13%	0%

distorted by the projector-camera channel effects. Third, creation attacks result in higher success rates than alteration attacks do because in alteration attacks there are benign images in the background, encouraging the classifier to make correct classifications. We will address these issues in the next section, so as to increase the overall attack success rate.

3.5 System-aware GhostImage Attacks

There are some limitations of the camera-aware attack introduced in the previous section. First, increasing distances results in lower success rates because the classifier cannot recognize the resulting low-resolution images. Second, there are large gaps between digital domain results and perception domain results, as channel effects (which cause the inconsistency between the intended pixels and the perceived pixels) are not taken into account. In this section, we resolve these limitations and improve GhostImage attacks' success rates by proposing a framework which consists of a channel model that predicts the pixels perceived by the camera, given the pixels as input to the projector, as well as an optimization formulation based on which the attacker can solve for optimal attack patterns that cause misclassification by the target classifier with high confidence.

3.5.1 Technical Challenges

First, the injected pixel values are often difficult to control as they exhibit randomness due to variability of the channel between the projector and the camera, thus the adversary is not able to manipulate each pixel deterministically. Second, to achieve optimal results, the attacker needs to precisely predict the projected and perceived pixels, thus channel effects must be modeled in an end-to-end manner, i.e., considering not only the physical channel (air propagation), but also the internal processes of the projector and the camera. Lastly, the resolution of attack patterns is limited by distances and projector lens (Eq. 3.7), thus the ghost patterns must be carefully designed to fit the resolution with few degrees of freedom.

3.5.2 System-aware Attack Overview

The system-aware attacker aims to find optimal patterns that can cause misclassification by the target classifier with high confidence by taking advantage of the non-robustness of the classifier [49]. We adopt an adversarial example-based optimization formulation into GhostImage attacks, in which the attacker tries to solve

$$\Delta^* = \arg\min_{\Delta} \quad \|\Delta\|_p + c \cdot \mathcal{L}(y, t, \theta), \tag{3.8}$$

where Δ is the digital attack pattern as input to the projector, y is the perceived image of the object of interest under attacks, t is the target class, and θ represents the targeted neural network. $\|\cdot\|_p$ is an ℓ_p-norm that measures the magnitude of a vector, and $\mathcal{L}$ is a loss function indicating how (un)successful Δ is. Here, we aim to minimize the power of the projector required for a successful attack, meanwhile maximizing the successful chance of attacks. The relative importance of these two objectives is balanced by a constant c. Sec. 3.5.4 details (3.8) in terms of how we handle Δ being a non-negative tensor that is also able to depict grid-style patterns in different resolutions.

More importantly, in (3.8) y is the final perceived image used as input to the classifier, which is estimated by our channel model in an end-to-end style (Fig. 3.10), in which δ [3] is the input to the projector, and y is the resulting image captured by the camera. The model can be formulated as

$$y = g\left(h_f(\Delta) + h_o(x)\right). \tag{3.9}$$

where $h_f(\Delta)$ is the ghost model that estimates the perceived adversarial pixel values in the ghost. For simplicity we let $h_o(x) = x$ because the attacker possesses same type of the camera so that x can be obtained a priori, and $g(\cdot)$ is the auto exposure control that adjusts the brightness. Sec. 3.5.3 introduces the derivation of (3.9).

Next, we will first present the channel model, and then formulate the optimization problem for finding the optimal adversarial ghost patterns.

[3] Different than Δ which is a $w \times h \times 3$ tensor, δ is a single pixel with dimension 3×1 for the convenience of the analysis.

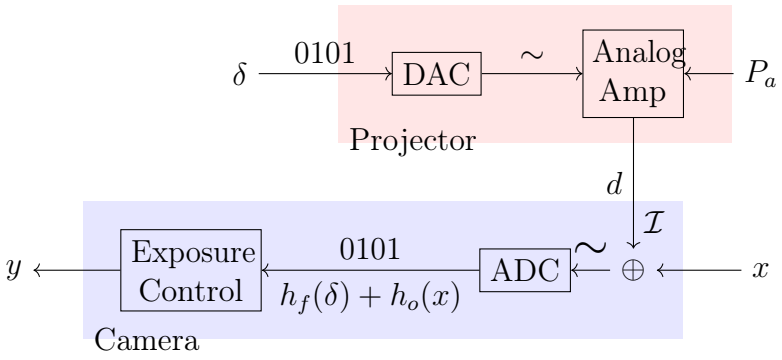


Figure 3.10: Projector-camera channel model

3.5.3 Projector-Camera Channel Model

We consider the projector to camera channel model (Fig. 3.10) in which δ is an RGB value the attacker wishes to project which is later converted to an analog color by the projector. The attacker can control the power (P_a) of the light source of the projector so that the luminescence can be adjusted. The targeted camera is situated at a distance of d, which captures the light coming from both the projector and reflected off the object (x). The illuminance received by the camera from the projector is denoted as $\mathcal{I}$. The camera converts analog signals into digital ones, based on which it adjusts its exposure, with the final, perceived RGB value being y. An ideal channel would yield $y = x + \delta$ but due to channel effects, we need to find a way to adjust the projected RGB value such that the perceived RGB value is as intended, i.e., to find the appropriate x given y.

3.5.3.1 Exposure control

As we discussed in Section 3.3.1, cameras are usually equipped with auto-exposure control, where according to the overall brightness of the image, the camera adjusts its exposure by changing the exposure time, or the size of its aperture, or both. We observed from our experiments that, as we increase the luminescence of the projector ($\mathcal{I}$), in the image the brightness of the object (x) decreases but the ghost (δ) does not decrease as much. Modeling such phenomena helps the attacker to precisely predict the perceived image. For the following, we will first find out how the illuminance $\mathcal{I}$ depends on δ and P_a (the normalized power of light bulb ranging from 0% to 100%), and then analyze how y depends on $\mathcal{I}$.

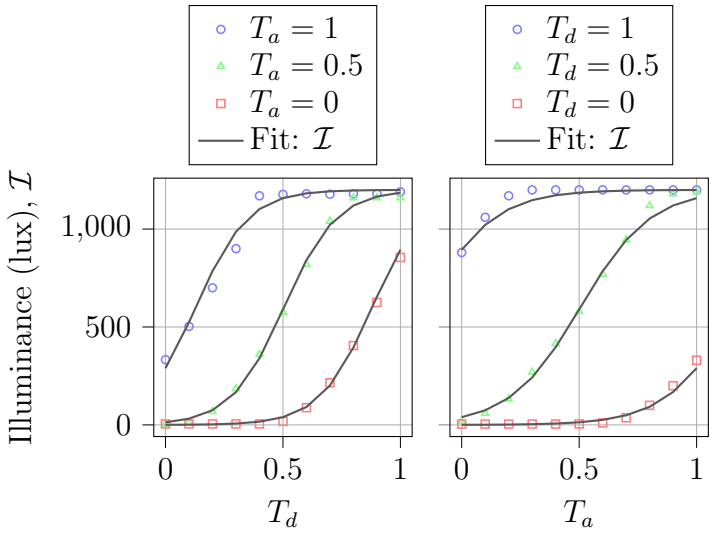


Figure 3.11: Illuminance depends on the RGB amplitude T_d, and the light bulb intensity T_a.

How does $\mathcal{I}$ depend on δ and P_a? We conducted a series of experiments, where $T_d = \|\delta\|_\infty = \max_i \delta_i$ and P_a were varied. We recorded the illuminance directly in front of the camera using an illuminance meter, with the projector one meter away. The results are plotted in Fig. 3.11, which shows that

$$\mathcal{I}(T_d, P_a, d) = \frac{c_d}{d^2} \cdot \frac{\mathcal{I}_{\max}}{1 + e^{-t}}, \tag{3.10}$$

where $t = a \times T_d + b \times P_a + c_t$, and a, b, c_d and c_t are constants derived from the data. $\mathcal{I}_{\max}$ is the maximum illuminance of the projector at a distance of one meter. Such a sigmoid-like function captures the luminescence saturation property of the projector hardware.

How does the perceived x depend on $\mathcal{I}$? In the same experiments we also recorded the RGB value of the ghost (δ) with a blackboard as background (in order to reduce ambient impacts), and a piece of white paper (x) that was also on the blackboard but did not overlay with the ghost. Their data are shown in Fig. 3.12, from which we can derive the *dimming ratio* that measures the change of exposure/brightness:

$$\gamma(\mathcal{I}) = \frac{\mathcal{I}_{\text{env}}}{\mathcal{I} + \mathcal{I}_{\text{env}}}, \tag{3.11}$$

where $\mathcal{I}_{\text{env}}$ is the ambient lighting condition in illuminance which differs from indoors to outdoors for instances. From this equation, we see that in an environment with static lighting condition, as the luminescence of the projector increases, the dimming ratio decreases, hence the objects become darker. With (3.11), the adversary is able to conduct real-time attacks by simply plugging in the momentary $\mathcal{I}_{\text{env}}$.

How does the perceived δ depend on $\mathcal{I}$? When $x = 0$, $\|y_f\| = \|y_f\|_\infty$ (the lower subplot of Fig. 3.12) depends on $\mathcal{I}$ in two ways:

$$\|y_f\|(\mathcal{I}) = \gamma(\mathcal{I}) \cdot \rho \cdot \mathcal{I}.$$

On one hand, the last term $\mathcal{I}$ increases the intensity of ghosts, but on the other hand the dimming ratio $\gamma(\mathcal{I})$ dims down ghost, whereby ρ is a trainable constant. With this, we can rewrite the perceived flare as

$$y_f = \|y_f\| H_c \frac{\delta}{\|\delta\|},$$

where H_c is the color calibration matrix to deal with color distortion, which will be discussed in Section 3.5.3.2. The term $1/\|\delta\|$ normalizes δ. In the end, we have the channel model

$$y = \gamma(\mathcal{I}) \left(\rho \mathcal{I} H_c \frac{\delta}{\|\delta\|} + x \right). \tag{3.12}$$

Compared to (3.9),

$$h_f(\delta) = \rho \mathcal{I} H_c \frac{\delta}{\|\delta\|}, \quad g(t) = \gamma(\mathcal{I}) t, \quad h_o(x) = x.$$

With (3.12), the attacker is able to predict how bright and what colors/pixel values the ghost and the object will be, given the projected pixels, the power of the projector, and the distance.

3.5.3.2 Color calibration

Considering a dark background (i.e., $x = 0$), (3.12) can be simplified as $y = \gamma(\mathcal{I}) \rho \mathcal{I} H_c \delta / \|\delta\|$, where H_c is a 3×3 matrix (as three color channels) that calibrates colors. Both y and δ are

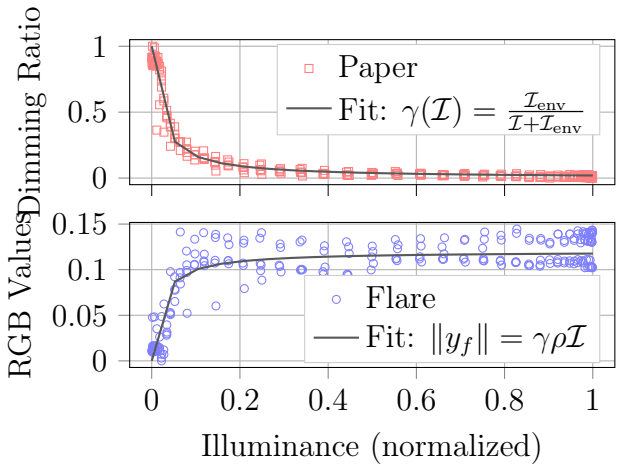


Figure 3.12: Perceived RGB values v.s. illuminance.

3×1 column vectors. H_c should be an identity matrix for an ideal channel, but due to the color-imperfection of both the projector and the camera, H_c needs to be learned from data. To simplify notations, we define corrected x and y as

$$\hat{x} = \frac{\delta}{\|\delta\|}, \quad \hat{y} = \frac{y}{\rho\mathcal{I}\gamma(\mathcal{I})},$$

so that we can write

$$\hat{y} = H_c\hat{x}.$$

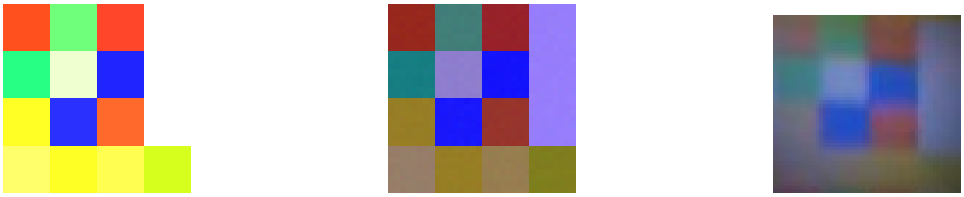
We did another set of experiments where we collected $n = 100$ pairs of $(\hat{x}, \hat{y})$ with dark background (to make $x = 0$), with δ being assigned randomly, and $P_a = 30\%$. We grouped them into X and Y:

$$X = \left[\hat{x}_1^\top, \hat{x}_2^\top, \cdots, \hat{x}_n^\top\right]^\top, \quad Y = \left[\hat{y}_1^\top, \hat{y}_2^\top, \cdots, \hat{y}_n^\top\right]^\top,$$

where both X and Y are $n \times 3$ matrices. We solve

$$\min_{H_c} \|Y - XH_c\|_2^2.$$

to obtain H_c, which is known as a non-homogeneous least square problem [143], and has a



(a) Input x_f **(b)** Estimated y **(c)** Actual y

Figure 3.13: An example of channel model prediction

closed-form solution:

$$H_c = \left((X^\top X)^{-1} X^\top Y\right)^\top.$$

Plugging H_c back to (3.12) completes our channel model.

3.5.3.3 Model validation

Fig. 3.13 demonstrates the accuracy of our channel model. In it the left image is the original input to the projector, the middle image is the estimated output from the camera based on our channel model (Eq. 3.12), and the image on the right is the actual image in a ghost captured by the camera. As can be seen, the difference between the actual and predicted is much less than the actual and original. While blurring effect is apparent in the actual y, we do not model it but the success rates are still high despite it. As we will see in Section 3.7, our channel model is general enough that once trained on one camera in one environment, it can be transferred to different environments and different cameras without retraining.

3.5.4 Optimal Adversarial Projection Patterns

In long-distance, low-resolution GhostImage attacks there are only a few pixels in the ghost area. A camera-aware attacker's strategy is to simply downsample attack images into low resolutions, but that does not result in high success rates. While (3.8) is abstract, for the rest of this subsection, we will progressively detail it and show how it can be solved in light of the channel model to improve attack success rates. We will start with the simplest case where adversarial perturbations are random noise (Sec. 3.5.1). Then, single-color ghosts will be introduced. Later, we will consider how to find semi-positive additive noise due to the fact

that superposition can only increase perceived light intensity but not decrease it. Finally, we examine the optimization problem to find optimal ghost patterns in grids at different resolutions.

3.5.4.1 Ghosts in random noise

Let us consider the simplest case first where the random noise Δ is drawn from one single Gaussian distribution for all three channels, i.e., $\Delta \sim \mathcal{N}(\mu, \sigma^2)$, where the size of Δ is $w \times h \times 3$ with w and h representing the width and height of the benign image x. This is because the values of each pixel that appear in the ghost area follow Gaussian distributions according to statistics obtained from our experiments.

The adversary needs to find μ and σ such that when Δ is added to the benign image x, the resulting image y will be classified as the target class t. That said, the logits value (Section 3.3.2) of the target class should be as high as possible compared with the logits values of other classes [19]. Such a difference is measured by the loss function $\mathcal{L}(y, t)$

$$\mathcal{L}(y,t) = \max\left\{-\kappa, \max_{i:i\neq t}\{\mathbb{E}[Z_i(y)]\} - \mathbb{E}[Z_t(y)]\right\}, \tag{3.13}$$

where $\mathbb{E}[Z_i(y)]$ is the expectation of logits values of Class i given the input y. Term $\max_{i:i\neq t}\{\mathbb{E}[Z_i(y)]\}$ is the highest expected logits value among all the classes except the target class t, while $\mathbb{E}[Z_t(y)]$ is the expected logits value of t. Here, κ controls the logits gap between $\max_{i:i\neq t}\{\mathbb{E}[Z_i(y)]\}$ and $\mathbb{E}[Z_t(y)]$; the larger the κ is, the more confident that Δ is successful. The attacker needs $\mathcal{L}$ as low as possible so that the neural network would classify y as Class t. Most importantly, y is computed based on our channel model (Eq. 3.12), so that the optimizer finds the optimal ghost patterns that are resistant to the channel effects. Unfortunately, due to the complexity of neural networks, the expectations of logits values $\mathbb{E}[Z_i(y)]$ are hard to be expressed analytically; we instead use Monte Carlo methods to approximate it:

$$\hat{\mathbb{E}}[Z_i(y)] = \frac{1}{T}\sum_{j=1}^{T} Z_i(y_j),$$

where T is the number of trials, and y_j is of the j-th trial.

Meanwhile, the adversary also needs to minimize the magnitude of Δ to reduce the attack power and noticeability, as well as its peak energy consumption, quantified by σ. The expectation of the magnitude of Δ is

$$\mathbb{E}[\|\Delta\|_p] = \mu n^{1/p}, \quad \text{with} \;\; n = 3wh. \tag{3.14}$$

Putting (3.13) and (3.14) together with a tunable constant c, we have our optimization problem for the simplest case

$$\begin{aligned} \mu^*, \sigma^* = \; & \arg\min_{\mu,\sigma} \quad \mathbb{E}[\|\Delta\|_p] + \sigma + c \cdot \mathcal{L}(y,t), \\ & \text{such that} \quad \sigma > \sigma_l, \end{aligned}$$

Here, σ_l is the lower bound of the standard deviation σ, meaning that the interference generator and the channel environment can provide random noise with the standard deviation of at least σ_l. When $\sigma_l = 0$, the adversary is able to manipulate pixels deterministically. Therefore, when we fix σ as σ_l in the optimization problem, the attack success rate when deploying μ^* would be the lower bound of the attack success rate. In other words, the adversary equipped with an attack setup that can produce noise with a lower variance (than σ_l^2) can carry out attacks with higher success rates. Therefore, we can simplify our formulation by removing the constraint about σ, so the optimization problem becomes

$$\mu^* = \arg\min_{\mu} \;\; \mathbb{E}[\|\Delta\|_p] + c \cdot \mathcal{L}(y,t). \tag{3.15}$$

For the rest of the chapter we will simply use σ to denote σ_l.

3.5.4.2 Ghosts in single-color

Since in (3.15) there is only one variable that the adversary is able to control, it is infeasible to launch a targeted attack with such few degrees of freedom. As a result, the adversary needs to manipulate each channel individually. That is, for each channel, there will be an independent distribution from which noise will be drawn. This is feasible because noise can appear in different colors in the ghost areas in which three channels are perturbed differently

when using projectors. Let us decompose Δ as $\Delta = [\Delta_R, \Delta_G, \Delta_B]$, where the dimension of $\Delta_{\{R,G,B\}}$ is $w \times h$, and they follow three independent Gaussian distributions

$$\Delta_R \sim \mathcal{N}\left(\mu_R, \sigma_R^2\right), \quad \Delta_G \sim \mathcal{N}\left(\mu_G, \sigma_G^2\right), \quad \Delta_B \sim \mathcal{N}\left(\mu_B, \sigma_B^2\right).$$

Here, $\mu_{\{R,G,B\}}$ and $\sigma_{\{R,G,B\}}$ are the means and the standard deviations (σ) of the three Gaussian distributions, respectively. The expectation of such Δ is then

$$\mathbb{E}[\|\Delta\|_p] = \left[\frac{n}{3}\left(\mu_R^p + \mu_G^p + \mu_B^p\right)\right]^{\frac{1}{p}}. \tag{3.16}$$

(3.14) is a special case of (3.16) when $\mu = \mu_R = \mu_G = \mu_B$. We denote $\boldsymbol{\mu} = [\mu_R, \mu_G, \mu_B]^\top$. Hence, similar to (3.15), we have the optimization problem for single-color perturbation [150]

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \quad \mathbb{E}[\|\Delta\|_p] + c \cdot \mathcal{L}(y, t), \tag{3.17}$$

by which the adversary finds $\boldsymbol{\mu}^*$ from which Δ is drawn.

3.5.4.3 Non-negative noise constraint

sometimes gives us a $\boldsymbol{\mu}^*$ of which one or more elements are negative, which is not able to be implemented in the perception domain because the perturbation introduced using ghost effects has to be non-negative, i.e., $\Delta \geq 0$; in other words, we can only increase the intensity of light.

There are mainly two approaches to achieve non-negativity. First, we can follow the change-of-variable method where we can use another variable that is non-negative. For example, we can simply clip Δ to be non-negative. We can also apply a function that outputs only positive values, e.g., *softplus* function. The second approach is to add a negativity penalty to the objective function to discourage the optimization algorithm from returning negative solutions.

Specifically, for the first approach, we can replace Δ with a non-negative one, Δ^+, then (3.17) becomes

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \quad \mathbb{E}[\|\Delta^+\|_p] + c \cdot \mathcal{L}(y, t), \tag{3.18}$$

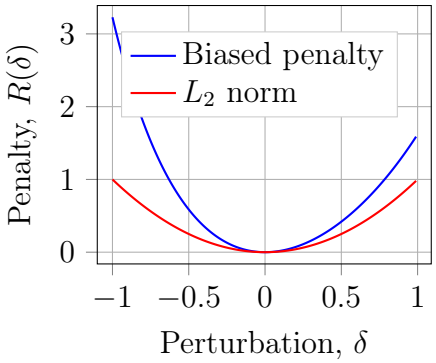


Figure 3.14: Biased penalty

where Δ^+ can be defined by one of the following two methods:

- Clipping: $\Delta^+ = \max(\Delta, 0)$.
- Softplus function: $\Delta^+ = \log(1 + e^{\Delta/4})$.

(3.17) must be solved with the constraint $\Delta \geq 0$ because the adversary can only increase the light. Rather than explicitly place a constraint in (3.17), we propose to punish negative values, by intruducing *biased penalty*

$$R(\Delta) = e^{-\alpha(\Delta-\omega)} + e^{\beta(\Delta-\omega)} - \eta, \tag{3.19}$$

where

$$\omega = \frac{\ln\alpha - \ln\beta}{\alpha+\beta}, \quad \eta = \left(\frac{\alpha}{\beta}\right)^{\frac{-\alpha}{\alpha+\beta}} + \left(\frac{\alpha}{\beta}\right)^{\frac{\beta}{\alpha+\beta}}.$$

Here ω is to center the global minimum at Δ being zero, and substracting η is to lower the minimum to be zero but it does not change the optimization results so we will omit it. An instance of (3.19) when $\alpha = 2$ and $\beta = 1$ is plotted in Fig. 3.14 in comparison with the L_2 norm. With the same absolute value, while the L_p norm treats positive perturbation and negative perturbation equally, the biased penalty function punishes the negative values more than the positive one, encouraging the optimization algorithm to find positive Δ. We adopt it into our optimization formulation

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \quad \mathbb{E}[R(\Delta)] + c \cdot \mathcal{L}(y, t), \tag{3.20}$$

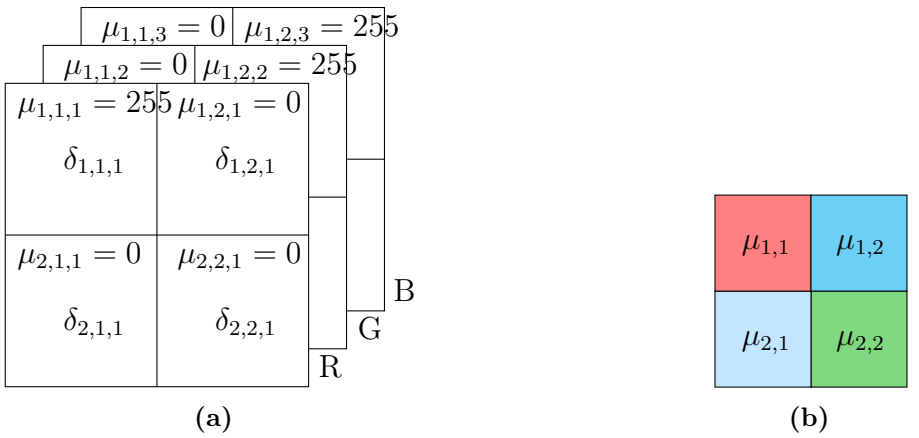


Figure 3.15: A grid pattern when $N_{row} = N_{col} = 2$ and $N_{chn} = 3$. (a) $\boldsymbol{\mu}$ is a three-dimensional matrix. (b) The resulting perturbation pattern.

and in the experiments we set $\alpha = 8$ and $\beta = 2$.

3.5.4.4 Ghost grids

Since projector's pixels are arranged in grids, the attack patterns are in grids as well, especially in lower resolutions. We enable Δ with patterns in different resolutions. Such a grid pattern $\boldsymbol{\Delta}$ can be composed of several blocks $\Delta_{i,j,k}$, i.e., $\Delta_{i,j,k} : \{1 \leq i \leq N_{\text{row}}, 1 \leq j \leq N_{\text{col}}, 1 \leq k \leq N_{\text{chn}}\}$ where N_{row}, N_{col} and N_{chn} is the number of rows, columns, and channels of a grid pattern, respectively, in terms of blocks. In a word, $\Delta_{i,j,k}$ is the perturbation block at i-th row, j-th column and k-th channel. A block $\Delta_{i,j,k}$ is a random matrix and its size is $\frac{w}{N_{\text{col}}} \times \frac{h}{N_{\text{row}}}$, so that the size of $\boldsymbol{\Delta}$ is still $w \times h \times 3$. Besides, the elements in the random matrix $\Delta_{i,j,k}$ is i.i.d. drawn from a Gaussian distribution, i.e., $\Delta_{i,j,k} \sim \mathcal{N}(\mu_{i,j,k}, \sigma^2)$.

The adversary finds the optimal grid pattern $\boldsymbol{\Delta}$ by solving

$$M^* = \arg\min_{M} \quad \mathbb{E}[R(\|\boldsymbol{\Delta}\|_p)] + c \cdot \mathcal{L}(y, t), \tag{3.21}$$

where $R(\boldsymbol{\Delta})$ is the softplus function to guarantee that the perturbation is always positive. $M = \{\mu_{i,j,k}\}$ is a tensor in shape $N_{\text{row}} \times N_{\text{col}} \times N_{\text{chn}}$. See Fig. 3.19a for some examples of adversarial grids in different resolutions. See Fig. 3.15a for an illustration of the dimensionality of $\mu_{i,j,k}$, and Fig. 3.15b for the resulting pattern in color.

The resolution of a grid pattern ($N_{row} \times N_{col}$) is set according to the pixel density of the

projector and the distance from the projector to the camera. The higher the pixel density of the projector is, the more complicated patterns the adversary would be able to inject to the imager. In our experiments, for example, our NEC projector [151] equipped with the Canon lens [3] is able to inject a grid pattern with 32×32 blocks when projector's resolution is 1024 by 768 pixels at a distance of one meter from the camera.

3.6 Spoofing Object Detection

So far, we have introduced a system-aware GhostImage attacker who manipulates only the image classifier. However, a more practical setting would be to compromise object detection models. You-only-look-once (YOLO) [1] is one example of such models. Due to its fast processing and high accuracy, YOLOv3 is being used in industrial self-driving car platforms, such as Apollo [15]. In this section, we use YOLOv3 as an example to demonstrate how a system-aware attacker is able to create an object that is misperceived by an object detection algorithm.

3.6.1 Challenges and Solutions

Compared with the functionality of image classification which is to assign a label for the entire image, object detection is to localize and classify (usually multiple) objects from the image. Due to such differences, it is still challenging to spoof YOLO even though we are able to spoof image classifiers. Here, we will discuss the technical challenges and how we address them. We still formulate an optimization problem similar to Eq. 3.8 but we need to update the objective function with additional constraints that capture the characteristics of YOLO and the projector-camera channel effects.

Shape and Contrast: According to our experiments, the shape of the injected patterns has a large impact on the attack effectiveness in the perception domain. For example, projecting a pattern in an octagon shape is more likely to be detected as a STOP sign by YOLO than a square-shaped pattern. Moreover, due to the channel effects (Sec. 3.5.3), the induced patterns do not typically have clear edges, thus indistinguishable from the background. In order to emphasize the shape, the edge of it needs to be of high contrast especially because

the blurring effect is inevitable. As a result, when formulating the optimization problem, the attacker needs to add a shape constraint that represents the target class, and another constraint that increases the brightness of the pixels on the edge so that in the perceived image the pattern can easily be distinguished from the background (Fig. 3.16c).

Image Context: Since the official YOLOv3 model that we evaluate was trained as a regression problem where all objects (including their bounding boxes and labels) of the input image were fed to the optimizer, it implicitly learned the context of the scene [63, 152], e.g., YOLO might have learnt that "there is typically a pole under a STOP sign although the bounding box only includes the octagon." In our experiments for inducing a STOP sign, placing a pole under the octagon pattern does increase the recognition confidence, which can be achieved by changing the shape constraint slightly (Fig. 3.16c). In addition, the background that surrounds the object also has an impact. In the case of our experiment setup, there is a blue-tinted ghost overlaying with the ghost that contains projected attack patterns, which distorts the attack pattern significantly. However, unless acquiring two or more light sources, the attacker would not be able to change the background (except its brightness). To overcome this, we instead update our channel model to simulate the second ghost as a constraint in the optimization (Fig. 3.16c), so that the optimizer finds optimal patterns anticipating the other ghost.

Robustness to Shift: As mentioned in Sec. 3.4.3, only a small portion of the projected pixels are captured by the camera, which is denoted as S_f (Fig. 3.5b). As the relative movement between the projector and the camera occurs, S_f shifts within the projection plane S. Although the adversary is able to track the camera, it would still be better if we can relax the requirement of precisely controlling the position of S_f. To improve the attack practicality, we can equip adversarial patterns $\mathbf{\Delta}$ with shift-invariance. This means, wherever S_f is within S, the ghost that contains S_f could always deceive YOLO. To achieve shift-invariance, we tile the pattern $\mathbf{\Delta}$ into a canvas and apply masks on the canvas, where different masks reveal/uncover random portions (equivalent to S_f in a fixed size) of the canvas (Fig. 3.16). Finally, we ask the optimizer to find a pattern $\mathbf{\Delta}$ that can succeed under all masks. Another benefit of masks is that we can easily vary the pattern shape in order to mimic different target classes.

Loss Function: Recall that YOLOv3 outputs a tensor in Shape $m \times m \times 3 \times 85$ while an image classifier outputs simply a vector (Sec. 3.3.2), we need to redesign our loss function $\mathcal{L}$ (Eq. 3.8) that measures how unsuccessful so far the pattern is. Generally, there are two terms that the attacker aims to manipulate. First, they aim to deceive YOLO that there is an object at (i, j, k) by increasing $P_x(i, j, k)$. Second, the attacker needs to make YOLO believe that the object is most likely of a specific class t by increasing $P_x^t(i, j, k)$. The attacker can either specify (i, j, k) at which the portion uncovered (by the mask) is, or simply using all permutations of (i, j, k). In our experience, the latter actually works faster because tensor slicing is not supported that well in TensorFlow.

3.6.2 Overview of Optimization

The formulation is the same as (3.8), with the loss function of creation attacks,

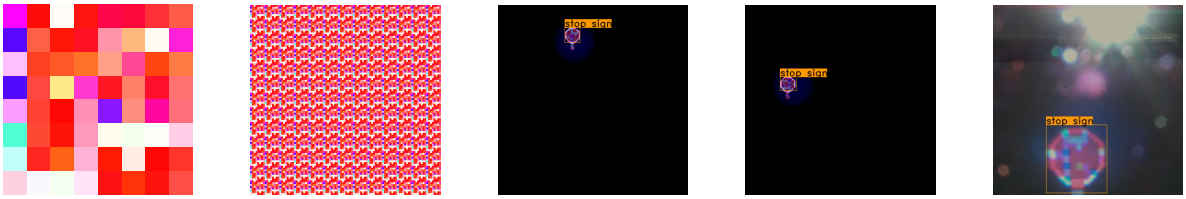
$$\mathcal{L}(y, t, \theta) = -\sum_{M_f \in \mathcal{M}} \sum_{i,j,k} \mathop{\mathbb{E}}_{y} \left[P_y(i, j, k) + P_y^t(i, j, k) \right], \tag{3.22}$$

where $\mathcal{M}$ is a set of masks that reveal different portions of the canvas $\mathbf{\Lambda}$ to achieve shift-invariance. In each mask M_f, the values are either 0, 1, or $b > 1$. 0 means that pixel is covered and 1 means uncovered; b's represent the edge of the shape which is made brighter. With this setting, when we take an element-wise product (denoted as $\otimes$) between a canvas and a mask, i.e., $\mathbf{\Lambda} \otimes M_f$, a pattern with an arbitrary shape and bright edges can be formed (Fig. 3.16c).

Moreover, y is derived by our channel model (Eq. 3.9), from the background x, the second ghost of interest, and the masked adversarial pattern $\mathbf{\Lambda} \otimes M_f$. We are taking the expectation of y here because y is a random variable. See Fig. 3.16 for some examples of Pattern $\mathbf{\Delta}$, Canvas $\mathbf{\Lambda}$, the masked canvas, and the second ghost of interest. Details of the formulation can be found in Sec. 3.6.3 from the supplementary material.

3.6.3 YOLOv3 as A Case Study

In this section, we formulate the optimization problem that the attacker solves to generate adversarial ghost patterns which are able to deceive an object detector. We demonstrate it



(a) Δ **(b) Λ** **(c)** **(d)** **(e)**

Figure 3.16: An example of shift-invariance. (a) A shift-invariant pattern in 8×8 resolution (zoomed in for a better presentation). (b) A canvas that is full of (a) in 416×416 resolution which is the input size of YOLOv3. (c)(d) YOLOv3 detects it as a STOP SIGN at different locations. The light, blue circle is the simulated second ghost. (e) A perceived ghost conveying the same pattern is detected as a STOP sign (zoomed in for a better presentation).

by applying creation attacks on YOLOv3. The high level formulation is the same as Eq. 3.8, but due to the differences between an image classifier and an object detector, we need to update the loss function $\mathcal{L}$ (Sec. 3.6.3.1). In terms of those technical challenges (Sec. 3.6.1) to enable the adversarial patterns with shift-invariance, and to shape them with distinguishable edges, we first generate a canvas, and then apply masks on it (Sec. 3.6.3.2). Finally, to simulate the second ghosts, we update our channel model from the perspective of the full frame (Sec. 3.6.3.3).

3.6.3.1 The Loss Function

In creation attacks, the attacker aims to make YOLO believe that there is an object of Class t in the image. There are two terms that they intend to manipulate. First, they aim to deceive YOLO that there is an object at (i, j, k) by increasing $P_x(i, j, k)$. Second, the attacker aims to make YOLO believe that the object is most likely of a specific class t by raising $P_x^t(i, j, k)$. In terms of the tuple (i, j, k), the attacker could of course specify it as the pixel locations of ghosts (which is achievable as explained in Sec. 3.4.2). However, to make the attack easier, and to enable the adversarial patterns with shift-invariance (introduced soon), we take all possible tuples into consideration by summing them up:

$$\mathcal{L}(y, t, \theta) = -\sum_{i,j,k} \mathop{\mathbb{E}}_{y} \left[P_y(i, j, k) + P_y^t(i, j, k) \right],$$

where y is the input image, t is the target class, and θ is the parameters of a pre-trained YOLOv3 model. We are taking the expectation of y here because y is a random variable. More importantly, y will be composed by our extended channel model, of the background x, the masked adversarial pattern (Sec. 3.6.3.2), and the second ghost that overlaps with the one that contains the adversarial pattern (Sec. 3.6.3.3). We take the sum of two confidences rather than the product because the former is more efficient with the optimizer according to our experiments.

3.6.3.2 Shift-invariance and Masks

As mentioned in Sec. 3.4.3, only a small portion of the projection pixels is captured by the camera, which is denoted as S_f (Fig. 3.5b). As the relative movement between the projector and the camera occurs, S_f shifts within the projection plane S. Although the adversary is able to track the camera, controlling the position of S_f still requires high precision. To make the attack easier to conduct, we propose to equip adversarial patterns $\mathbf{\Delta}$ with shift-invariance.

Specifically, the attacker intends to find a pattern $\mathbf{\Delta}$ that repeats itself in S, so that no matter where S_f is within S, the ghost that contains S_f would always be able to deceive YOLO. The solution that we propose is to tile a small pattern $\mathbf{\Delta}$ into a large canvas $\mathbf{\Lambda}$, which spans S. Also, we apply masks M_f on $\mathbf{\Lambda}$ which simulates different locations of S_f in S (Fig. 3.16).

Formally, let us denote the canvas as $\mathbf{\Lambda}$ that is composed of $N_x \times N_y$ identical adversarial patterns $\mathbf{\Delta}$'s, i.e.,

$$\mathbf{\Lambda} = \{\Lambda_{i,j} = \mathbf{\Delta}, \forall i \in [1, N_x], j \in [1, N_y]\}.$$

N_x and N_y are chosen in a way that the size of $\mathbf{\Lambda}$ is the same as YOLOv3's input size (e.g., 416×416). A mask M_f is in the same size of $\mathbf{\Lambda}$. The majority of the mask values are zeros, but a small part of it are ones, representing S_f. Furthermore, on the edge of S_f, the mask values are b's which are greater than one, meaning that the pixels on the edge are made brighter than the rest. The pattern is in higher contrast when b gets larger. To create such masks, we simply use the OpenCV library to draw contours in different shapes (filled or not).

Let us define the center of the non-zero part of M_f as $C_f = (x_f, y_f)$. We will create a set of

1	2	1	2
3	4	3	4
1	2	1	2
3	4	3	4

Figure 3.17: A 2×2 pattern (1234) is tiled $N_x \times N_y$ times where $N_x = N_y = 2$. When a mask uncovers the grey part, it is equivalent to shifting the original pattern 1234 upwards by one and leftwards by one, cyclically.

masks, $\mathcal{M}$, that contains masks with varying centers (x_f, y_f) to make sure that, when taking the element-wise product between the canvas $\mathbf{\Lambda}$ and different masks, different portions of the canvas get uncovered, which is equivalent to shifting $\mathbf{\Delta}$ cyclically (Fig. 3.17).

In terms of creation attacks, for example, the loss function $\mathcal{L}(y, t, \theta)$ becomes

$$- \sum_{M_f \in \mathcal{M}} \sum_{i,j,k} \mathop{\mathbb{E}}_{y} \left[P_y(i,j,k) + P_y^t(i,j,k) \right],$$

where y is composed by our channel model (Eq. 3.9), of the background x, and the masked adversarial pattern $\mathbf{\Lambda} \otimes M_f$, in which $\otimes$ means element-wise product.

3.6.3.3 Extended Channel Model: The Second Ghost

As we mentioned in Sec. 3.6.1, the background that surrounds the attack pattern has an impact on YOLO's decision. In the case of our projector-camera setup, there is a blue-tinted ghost that often overlays with the ghost that carries the attack pattern, which causes inconsistency between the digital patterns and the perceived ones in terms of colors, shapes and contrast. To address this, we propose to extend our channel model from Sec. 3.5.3, by considering the second ghost into our optimization process, so that the optimizer finds a pattern that deceives YOLO even when it is distorted by the second ghost.

Similar to the process in Sec. 3.4.2, we collected and computed the ratio r_2 of the second ghost, along with the ratio r_1 of the main ghost. Then, given the center of the main ghost (which is randomly picked when generating a mask M_f), $C_1 = (x_f, y_f)$, the center of the second ghost is then

$$C_2 = \left(\frac{r_2}{r_1} x_f, \frac{r_2}{r_1} y_f \right).$$

Table 3.2: Neural network architecture for LISA dataset

Layer Type	Model
ReLU convolution	64 8×8-filters
ReLU convolution	128 6×6-filters
ReLU convolution	128 5×5-filters
ReLU Fully-connected	256
Softmax	8

In terms of color, we simply take the average RGB values of our attack pattern $\mathbf{\Delta}$, and choose a simple filled circle as the shape. To further simulate it, we also apply gradient filter on its edge so that it fades out smoothly (Fig. 3.16c).

3.7 System-aware Attack Evaluation

In this section, we consider camera-based image classification systems, as used in self-driving vehicles and surveillance systems, to illustrate the potential impact of our attacks. We present proof-of-concept system-aware attacks in terms of *attack effectiveness*, namely how well system-aware attacks perform in the same setup as camera-aware attacks (Section 3.4.5), and *attack robustness*, namely how well system-aware attacks are when being evaluated in different setups.

We will again use attack success rates as our metric. We used the Adam Optimizer [153] to solve our optimization problems. There are two sets of results: *Emulation results* refer to the classification results on emulated, combined images of benign images and attack patterns using our channel model (Equation 3.12). Emulation helps us conduct scalable and fast evaluations of GhostImage attacks before conducting real-world experiments[4]. *Experimental results* refer to the classification results on the images that are actually captured by the victim cameras when the projector is on.

The architectures of two neural networks are shown in Tables 3.2 and 3.4 and their hyper-parameters are in Table 3.5. The balanced LISA dataset is also detailed in Table 3.3.

[4]Source code is at https://github.com/harry1993/ghostimage

Table 3.3: Balanced LISA dataset

Index	Sign Name	Quantity
0	Added Lane	80
1	Merge	80
2	Pedestrian Crossing	80
3	School	77
4	Signal Ahead	80
5	Stop	80
6	Stop Ahead	80
7	Yield	80

Table 3.4: Neural network architecture for CIFAR-10 dataset

Layer Type	Model
ReLU convolution	64 8×8-filters
ReLU convolution	128 6×6-filters
ReLU convolution	128 5×5-filters
ReLU convolution	64 3×3-filters
Max Pooling	2×2
ReLU convolution	128 3×3-filters
ReLU convolution	128 3×3-filters
Max Pooling	2×2
ReLU Fully-connected	256
ReLU Fully-connected	256
Softmax	10

3.7.1 Attack Effectiveness

To compare with camera-aware attacks, system-aware attacks are evaluated in a similar procedure, targeting a camera-based object classification system with the LISA dataset and its classifier. The system uses an Aptina MT9M034 camera [145] in an in-lab environment.

3.7.1.1 Creation attacks

For emulated creation attacks, all distances (or all resolutions) yield attack success rates of 100% (Fig. 3.18), which means that our optimization problem is easy to solve. In terms of computational overhead, we need roughly 30 s per image at 2×2-resolution, and 10 s at 4×4 or above (because of more degrees of freedom) using an NVIDIA Tesla P100 [154].

Table 3.5: Training hyper-parameters

Parameter	Value
Learning Rate	0.1
Momentum	0.9
Dropout	0.5
Batch Size	128
Epochs	50

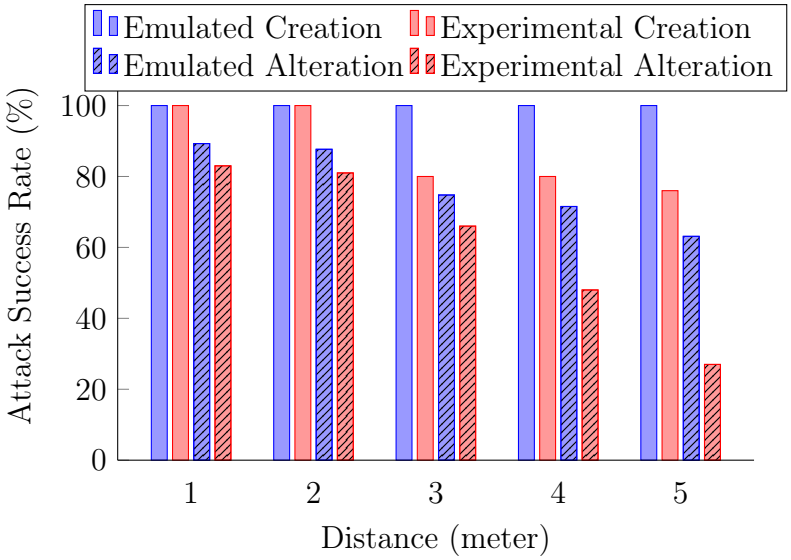


Figure 3.18: System-aware creation and alteration

Fig. 3.19a shows examples of emulated attack patterns for creation attacks, along with the images of real signs on the top. Interestingly, high-resolution shapes do look like real signs. For example, we can see two vertical bars for ADDEDLANE, and also we can see a circle at the middle south for STOPAHEAD, etc. These results are consistent with the ones from the MNIST dataset [50] where we could also roughly observe the shapes of digits. Secondly, they are blue tinted because our channel model suggests that ghosts tend to be blue, thus the optimizer is trying to find "blue" attack patterns that are able to deceive the classifier.

Interestingly, the all k resulting patterns of solving the optimization problem targeting one class from k different (random) starting points look similar to the ones shown in Fig. 3.19a. However, CIFAR-10 [155] and ImageNet [156] yield much different results: those patterns look rather random compared to the results from LISA or MNIST. The reason might be that in CIFAR-10, images in the same category are still very different, such as two different cats,

but in LISA, two images of STOP signs do not look as different as two cats.

For the experimental results of creation attacks, we see that as distances increase, success rates decrease a little (Fig. 3.18), but much better than the camera-aware attacks (Table 3.1), because the optimization formulation helped find those optimal attack patterns with high confidence.

3.7.1.2 Alteration attacks

The emulated and experimental results of alteration attacks are shown in Fig. 3.18. Compared with creation attacks, alteration attacks perform a bit worse, especially for large distances (three meters or further). This is because the classifier also "sees" the benign image in the background and tends to classify the entire image as the benign class. Moreover, the alignment of attack patterns and the benign signs is imperfect. However, when we compare Fig. 3.18 with Table 3.1 for camera-aware alteration attacks, we can see large improvements. Fig. 3.19b provides an example of system-aware alteration attacks in the perception domain, which were trying to alter the (printed) STOP sign into other signs: they look "blue" as the channel model predicted. The fifth column is not showing as it is STOP.

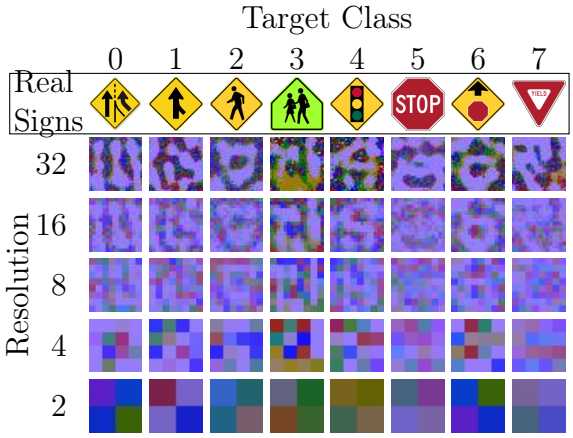
3.7.2 Attack Robustness

We evaluate the robustness of our attacks in terms of different datasets, environments, and cameras.

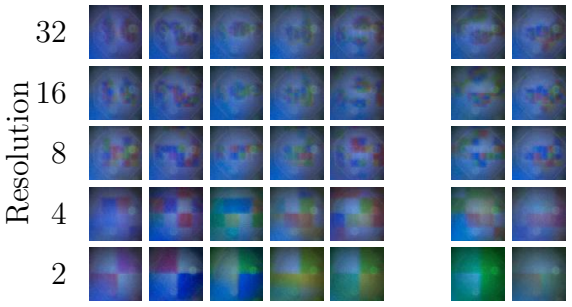
3.7.2.1 Different image datasets

Here we evaluate our system-aware attacks on two other datasets, CIFAR-10 [155] and ImageNet [156], by emulation only because previous results show that our attack emulation yields similar success rates as experimental results.

CIFAR-10 The network architecture and model hyper parameters are identical to [19]. The network was trained with the distillation defense [39] so that we can evaluate the robustness of our attacks in terms of adversarial defenses. A classification accuracy of 80% was achieved. The evaluation procedure is similar to Sec. 3.4.5.2. Results are shown in Fig. 3.20. The



(a) Emulated creation attacks



(b) Experimental alteration attacks

Figure 3.19: System-aware attack pattern examples.

overall trend is similar to the LISA dataset, but the success rates are significantly higher. The reason might still be the large variation within one class (Section 3.7.1.1), so that the CIFAR-10 classifier is not as sure about one class as the LISA classifier is, hence is more vulnerable to GhostImage attacks.

Transferability: Adversarial examples based on the distillation model are able to deceive the model trained without distillation, with 3.2% lower attack success rates on average. Such strong transferability is because we do not bound the noise norm. According to Carlini and Wagner [19], intense noise (with a larger κ) typically result in higher transferability.

ImageNet We used a pre-trained Inception V3 neural network [157] for the ImageNet dataset to evaluate the attack robustness against large networks. Since the pre-trained network can recognize 1000 classes, we did not iterate all of them [19]. Instead, for alteration attacks, we randomly picked ten benign images from the validation set, and twenty random

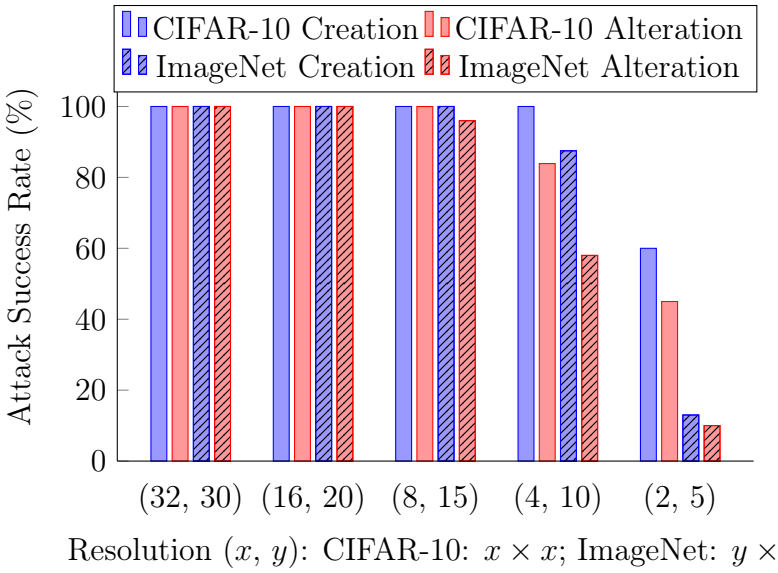


Figure 3.20: System-aware attacks on CIFAR-10 and ImageNet

target classes, while for creation attacks, the "benign" images were purely black. Results are given in Fig. 3.20.

For high resolutions ($\geq 15 \times 15$), the attack success rates were nearly 100%. But as soon as the resolutions went down to 10×10 or below, the rates decreased sharply. The reason might be that in order to mount successful *targeted* attacks on a 1000-class image classifier, a large number of degrees of freedom are required. 10×10 or lower resolutions plus three color channels might not be enough to accomplish targeted attacks. To verify this, we also evaluated untargeted alteration attacks on ImageNet. Results show that when the resolutions are 1×1 or 2×2, the success rates are 50% or 80%, respectively. But as soon as the resolutions go to 3×3 or above, the success rates reach 100%. Lastly, similar to CIFAR-10, system-aware attacks on ImageNet were more successful than on LISA, because of the high variation within one class.

3.7.2.2 Outdoor experiments

In order to evaluate system-aware attacks in a real-world environment, we also conducted experiments outdoor (Fig. 3.21), where the camera was put on the hood of a vehicle that was about to pass an intersection with a STOP sign. The attacker's projector was placed on the right curb, and it was about four meters away from the camera. The experiments

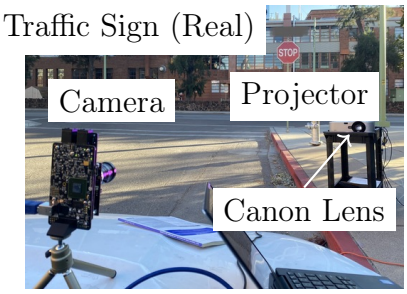


Figure 3.21: Outdoor experiment setup

Table 3.6: Outdoor alteration attack success rates

Success rates of	Noon	Dusk	Night
Overall	0%	51%	42.9%
STOP → YIELD	0%	100%	100%
STOP → ADDEDLANE	0%	100%	100%
STOP → PEDESTRIAN	0%	100%	100%

were done at noon, at dusk and at night (with the vehicle's front lights on) to examine the effects of ambient light on attack efficacy. The illuminances were 4×10^4 lx, 4×10^3 lx, and 30 lx, respectively. The experiments at noon were unsuccessful due to the strong sunlight. Although more powerful projectors [158] could be acquired, we argue that a typical projector is effective in dimmer environments (e.g., cloudy days, at dawn, dusk, and night, or urban areas where buildings cause shades), which accounts for more than half of a day. See Sec. 3.8.1 for more discussion on ambient lighting conditions.

Results (Tab. 3.6) of the other cases show that the success rates are 30% lower than our in-lab experiments (the four-meter case from Fig. 3.18), because we used our in-lab channel model directly in the road experiments without retraining it, and also the environmental conditions are more unpredictable. Moreover, the attack rates on altering some classes (e.g., the STOP sign) into three other signs (e.g., YIELD) were 100%, which is critical as an attacker can easily prevent an autonomous vehicle from stopping at a STOP sign.

3.7.2.3 Different cameras

Previously, we conducted GhostImage attacks on Aptina MT9M034 camera [145] designed for autonomous driving. Here, we evaluate two other cameras, an Aptina MT9V034 [159] with

Table 3.7: Channel model parameter examples

Description	Symbol	Value
Throwing ratio	r_{throw}	20
Physical size of ghosts	S_f	0.0156 cm^2
Projection resolution	P_O	1024×768
Flare booster	ρ	30
Bulb intensity	T_a	$[0, 1]$
Ambient illuminance	$\mathcal{I}_{env}$	300 lx (indoor)
Projector ill.	$\mathcal{I}$	400 lx (at 1 m)
Projector max ill.	$\mathcal{I}_{\max}$	1200 lx (at 1 m)
Camera matrix	M	See below
Color calibration matrix	H_c	See below
In Equation 3.10	a	8.9
In Equation 3.10	b	6.7
In Equation 3.10	c_t	-7.8
In Equation 3.10	c_d	0.25

a simpler lens design, and a Ring indoor security camera [148] for surveillance applications.

Aptina MT9V034 We mounted system-aware creation attacks against the same camera-based object classification system as in Section 3.7.1 but we replaced the camera with the Aptina MT9V034 camera. Since this camera has a smaller aperture size and also a simpler lens design than Aptina MT9M034, for a distance of one meter, only 16×16-resolution attack patterns could be achieved (previously we had 32×32 at one meter). We did not train a new channel model for this camera, so the attack success rate at one meter was only 75%, which is 25% lower than the Aptina MT9M034 camera. As the distances increased up to four meters, creation attacks yielded success rates as 46.25%, 33.75%, and 12.5%, respectively. Another reason why the overall success rate was lower is that even though the data sheet of Aptina MT9V034 [159] states that the camera also has the auto exposure control feature, we could not enable the feature in our experiments. In other words, system-aware creation attacks did not benefit from the exposure control. This, on the other hand, indicates the robustness of GhostImage attacks: Even without taking advantage of exposure control, the attacks were still effective, with attack success rates as high as 75%.

Ring indoor security camera We tested GhostImage untargeted attacks against a Ring indoor security camera [148] on the ImageNet dataset. To demonstrate that our attacks can be applied to surveillance scenarios, we assume the camera would issue an intrusion warning if a specific object type [160] is detected by the Inception V3 neural network [157]. The attacker's goal is to change an object for an intruder class to a non-intruder class. However, we could not find "human", "person" or "people", etc. in the output classes, we instead used five human related items (such as sunglasses) as the benign classes. We found six images from the validation set of ImageNet, of which top-1 classification results are one of those five benign classes. The six images were displayed on a monitor. For each benign image, we calculated ten alternative 3×3 attack patterns (the highest resolution at one meter by the Ring camera). Results show that for all six benign images, system-aware attacks achieved untargeted attack success rates of 100% (Table 3.8).

3.7.3 YOLOv3

So far, we have tested GhostImage system-aware attacks against image classifiers trained with different datasets in different architectures. Here, we test system-aware attacks against YOLOv3 as a proof-of-concept experiments to demonstrate how an object objector can be fooled to detect an object created by an attacker.

3.7.3.1 Evaluation Methodology

The attack setup is the same as depicted in Sec. 3.4.4: We use the NEC projector [2] with a replaced lens [3], and the victim camera is Aptina MT9M034 [145] that is 1 m away. We

Table 3.8: GhostImage untargeted alteration attacks against Ring camera on ImageNet dataset in perception domain

Index	Benign Class	Rate	Common Prediction
19992	fur boat	100%	geyser, parachute
21539	sunglasses	100%	screen, microwave
22285	sunglasses	100%	plastic bag, geyser
31664	sarong	100%	jellyfish, plastic bag
2849	sweatshirt	100%	laptop, candle
26236	puncho	100%	table lamp

Table 3.9: Success rates of creation attacks against YOLOv3 in Resolutions 8×8, 16×16, and 32×32 at one meter.

Label	Emulated			Experimental		
	8	16	32	8	16	32
stop sign	100%	100%	100%	82%	53%	13%
traffic light	100%	100%	100%	86%	50%	38%
fire hydrant	100%	100%	100%	33%	13%	13%
car	100%	100%	100%	87%	38%	13%
tv monitor	90%	100%	100%	88%	63%	50%

use a TensorFlow implementation [161] of the YOLOv3 architecture with the official network parameters [1], which was pre-trained on the COCO dataset [162] with 80 classes.

We iterate the attack resolution from 8×8 [5] due to the large capacity of YOLOv3. The attack success rates are calculated as the number of images that contain one or more objects with the targeted labels divided by the total number of images. For each label, we manually design the shape of the pattern based on the icons shown in the COCO dataset website [162] as they represent the typical shapes of differing classes. Since the camera outputs images in Resolution 2592×1944 [6] while YOLOv3 accepts 416×416 only, we first crop them to 1944×1944 and then downsample them to 416×416 in order for less distortion compared with direct downsampling.

Results Table 3.10 lists the attack success rates of creation attacks against YOLOv3 on traffic-related classes mainly. Firstly, the average success rates on cars is the relatively lower than other traffic-related labels, maybe because in the dataset, the variation of cars is large, meanwhile stop signs and traffic lights are typically the same in most of the training images. In addition, the success rate of fire hydrants is also low because fire hydrants are generally darker than their background while the ghost patterns are brighter than their background.

Secondly, traffic lights yield the highest success rates among other traffic-related labels, probably because the texture of ghost patterns is similar to how a (led array-like) light source would look like from a camera. After all, the ghosts are induced by a projector that is also one type of light sources. To verify this, we also induce TV monitors, yet another type of light

[5] The projected resolution
[6] The camera pixel resolution

Table 3.10: GhostImage creation attacks against YOLOv3: Success rates on traffic-related classes.

Resolution	person	bicycle	car	motorbike	bus	truck	traffic light	stop sign
32×32	100%	100%	100%	100%	100%	100%	100%	100%
16×16	100%	100%	100%	100%	100%	100%	100%	100%
8×8	100%	90%	100%	100%	100%	100%	93%	100%
4×4	100%	0%	100%	100%	100%	100%	63%	93%
2×2	100%	0%	100%	100%	100%	100%	20%	66%

sources; results show that it is indeed as successful as the traffic lights from the experimental evaluation, but the emulated success rate is slightly lower because the emulated patterns do not contain the light-source-style texture.

Thirdly, different than the results from image classification, the attacks do not benefit from high projected resolutions. Instead, as the resolution goes higher, the success rate becomes lower. This is because in the full-resolution images (1944×1944), a typical ghost pattern only occupies 320×320; when downsampling a 1944×1944 image to 416×416, the ghost pattern becomes only 68×68 with also noticeable blur effect. As a result, the projection pixels cannot be clearly seen in the downsampled images. To increase the attack success rate in this case, the attacker may choose to mount the attack at relatively large distances (which is actually stealthier), or be opportunistic for a camera system with large ghosts.

Also, we find out that the fading rate r_f has a great impact on the attack on YOLOv3, which implies that the edge detection of YOLOv3 is weak especially when edge is not clear, i.e., fading out rather than cutting out. As the fading rate decreases, the attack success rate decreases. Even for those patterns that are able to be recognized by YOLOv3, when they are applied a lower fading rate, they are no longer be able to do so.

3.8 Discussion

In this section, we discuss practical challenges to GhostImage attacks, speculate as to effective countermeasures.

3.8.1 Practicality of GhostImage Attacks

Moving targets and alignment: The overlap of ghosts and objects of interest in images must be nearly complete for the attacks to succeed. In the cases of a moving camera (e.g.,

one mounted to a vehicle), the attacker needs to be able to accurately track the movement of the targeted camera, otherwise the attacker can only sporadically inject ghosts. Note that the larger the aperture of a camera the larger the ghost(s); additionally, due to the increase in reflection paths, the greater the number of lens in the camera the greater number of ghosts, whose pixel coordinates can be calculated beforehand (Sec. 3.4.2). Although aiming (or tracking) moving targets is generally challenging in remote sensor attacks (e.g., the AdvLiDAR attack [80] assumes the attacker can achieve this via camera-based object detection and tracking), existing works [27, 138] have demonstrated the feasibility of tracking cameras and then neutralizing them. This chapter's main goal is to propose a new category of camera attacks, which enables an attacker to inject arbitrary patterns.

Conspicuousness: The light bursts around the light source in Figures 3.1 and 3.9 may raise stealthiness concerns about our attacks. However, according to our analysis in Sec. 3.4.2, such bursts can actually be eliminated because the light source can be outside of view [144]. Even the light source has to be in the frame (due to the lens configuration), we argue that a camera-based object classification system used in autonomous systems generally make decisions without human input (for example, in a Waymo self-driving taxi [17], no human driver is required). Additionally, the attack beam is so concentrated that only the victim camera can observe it while other human-beings (e.g., pedestrians) cannot (Fig. 3.21). Finally, the light source only needs to be on for a short amount of time, as a few tampered frames can cause incorrect actions [86].

Projectors, lenses, and attack distances: Based on our model (Eq. 3.11) and experiments (Tab. 3.7), the illuminance on the camera from the projector would better be 4/3 of the part from ambient illuminance (to achieve a success rate of 100%). Since $\text{Illuminance} \propto \text{Luminance} \cdot r_{\text{throw}}^2/d^2$, in order to carry out an attack during sunny days (typically with Illuminance $40 \times 10^3\,\text{lx}$), a typical projector (e.g., [163] with Luminance $9 \times 10^3\,\text{lm}$) should work with a telephoto lens [147] (with a throwing radio 100) at a distance of one meter. For longer distances or brighter backgrounds, one can either acquire a more powerful projector (e.g., [158] with $75 \times 10^3\,\text{lm}$), or combine multiple lenses to achieve much larger throwing ratios (e.g., two Optela lenses [147] yield 200, etc.), or both.

Ghost effect dependence: There are several challenges an attacker needs to overcome to

launch GhostImage attacks. First, the attacks rely largely on ghost effects; if ghosts cannot be induced, or if they are not significant enough, the attacks might be infeasible against a given camera (lens). However, this is unlikely because these effects occur in most cameras (e.g., Apple iPhones [164, 165]). Moreover, no "flare-free" lens exists to the best of our knowledge (even with anti-glare coatings). In fact, flare/ghost effects are desirable in photography [166] and filmography [167], which means camera manufacturers probably will not try to eliminate such artifacts. In addition, if ghost effects are unavailable to the attacker there are other optics effects available, such as blooming effects [168], that can also be leveraged to produce GhostImage-like attacks.

Knowledge of the targeted system: We assume that both types of attackers know about the camera matrix M_c and color calibration matrix H_c. We note that the attacks can still be *effective* without such knowledge but with it the attacks can be more *efficient*. For example, the attacker may choose to lower their attack success expectation but the probability of successful attack may still be too high for potential victims to bear (e.g., a success rate of only 10% might be unacceptable for reasons of safety in automated vehicles). This challenge can be largely eliminated if the attacker is able to purchase a camera of the same, or similar, model as used in the targeted system and use it to derive the matrices. Although the duplicate camera may not be exactly the same to the target one, the channel model would still be in the same form with approximate, probably fine-tuned parameters (via retraining), thanks to the generality of our channel model. Lastly, assuming white-box knowledge on sensors is widely adopted and accepted in the literature, e.g., the AdvLiDAR attack [80]. Also, we assume white-box attacks on the neural network, though this assumption can be eliminated by leveraging the transferability of adversarial examples [56, 169].

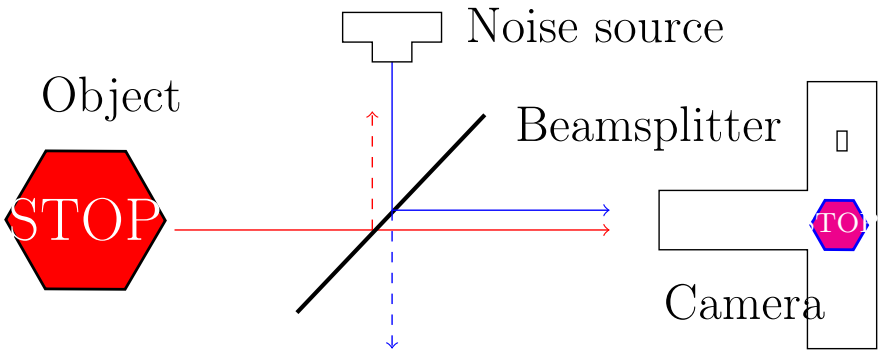
Object detection: We have assumed that the object detector can crop out the region of the image which contains the projected ghost pattern(s). Though it cannot be guaranteed that an object detector will automatically include the ghost patterns, we note that a GhostImage attacker could design ghost patterns that cause an object detector to include them [63, 170] and, at the same time, the cropped image would fool the subsequent object classifier.

Attack Variations: Should ghost effect be not available, we investigated alternative strategies that still allow an attacker to cause misclassificaiton of objects of interest. For

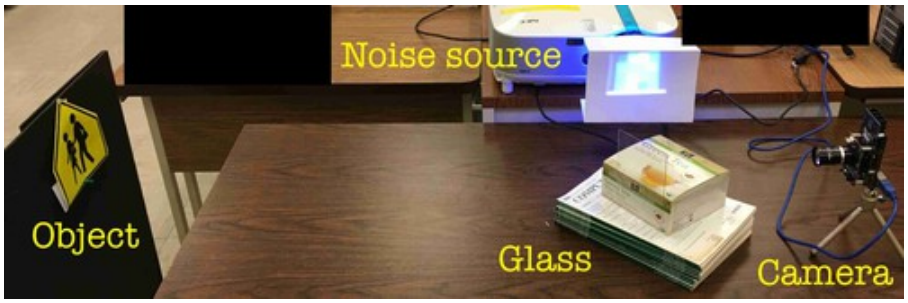
example, using a beamsplitter that merges two light beams coming from two directions: one is the light reflected from the object the attacker wishes to obscure and the other is the light from the projector. The merged light beams enter the targeted camera as a superposition of the original object and the adversarial pattern, with the resulting image able to fool the classifier. Finally, while projectors provide an attacker with the greatest control over adversarial patterns, and hence the ability to spoof complex objects, we have found that RGB lasers [171] can be used at greater distances to spoof simple objects. It may also be possible for an attacker to use multi-laser systems, or even flashlights [172], to create complex patterns akin to the decorative lights displayed on Christmas trees.

See Figure 3.22a for a diagram of this method, where a beamsplitter is used to merge two light beams coming from two directions. The light beams coming from the object (marked in red) get reflected and transmitted, i.e., the beamsplitter does not obscure the object. The transmitted portions go into the camera, forming an image of the object. The light beams from the light noise source (marked in blue) also get reflected and transmitted. The reflected portions (instead of the transmitted portions) are captured by the camera, forming an image of the noise. Two images overlap as a potential adversarial example, depicted as a small magenta (red plus blue) stop sign in the camera.

An in-lab experimental setup is shown in Figure 3.22b. In this setup, we used the NEC projector [151] as the light noise source. Instead of using an expensive beamsplitter, we found out that a single piece of glass could also function like a beamsplitter. We placed a piece of white paper in front of the projector's lens to reduce the projected image size (otherwise the projected image becomes too large even within a small throwing distance). This does not change the attack plausibility because the imager can clearly capture the noise pattern on the paper. These elements were placed in a way that the noise image (from the paper) would overlap with the image of the object from the view of the camera. A misclassification matrix is shown in Figure 3.23 where an overall attack success rate of 55% was achieved. Please see our technical report [173] for more details.



(a) Noise injection with a beam-splitter/glass



(b) Experiment setup with a piece of glass.

Figure 3.22: The beamsplitter method setup.

3.8.2 Countermeasures

The most straightforward countermeasure to GhostImage attacks is flare elimination, either by using a lens hood or through flare detection. Lens hoods are generally not favored as they reduce the angle of view of the camera, which is unacceptable for many autonomous vehicle and surveillance applications. Adversarial flare detection is challenging as they are typically transparent [174], and hard to be distinguished from natural ghosts.

A complementary line of defense would be to make neural networks themselves robust to GhostImage attacks. Existing approaches against adversarial examples (e.g., [39–41, 72], etc.) are ill-suited for this task, however, as GhostImage attacks do not necessarily follow the constraints placed on traditional adversarial examples in that perturbations do not have to be bounded within a small norm, meanwhile these defenses were not designed for arbitrarily large perturbations. As this work mainly focuses on sensor attacks, we leave the validation of defenses as future work [21, 60, 63, 80].

Another complementary approach of defense is to exploit prior knowledge, such as GPS

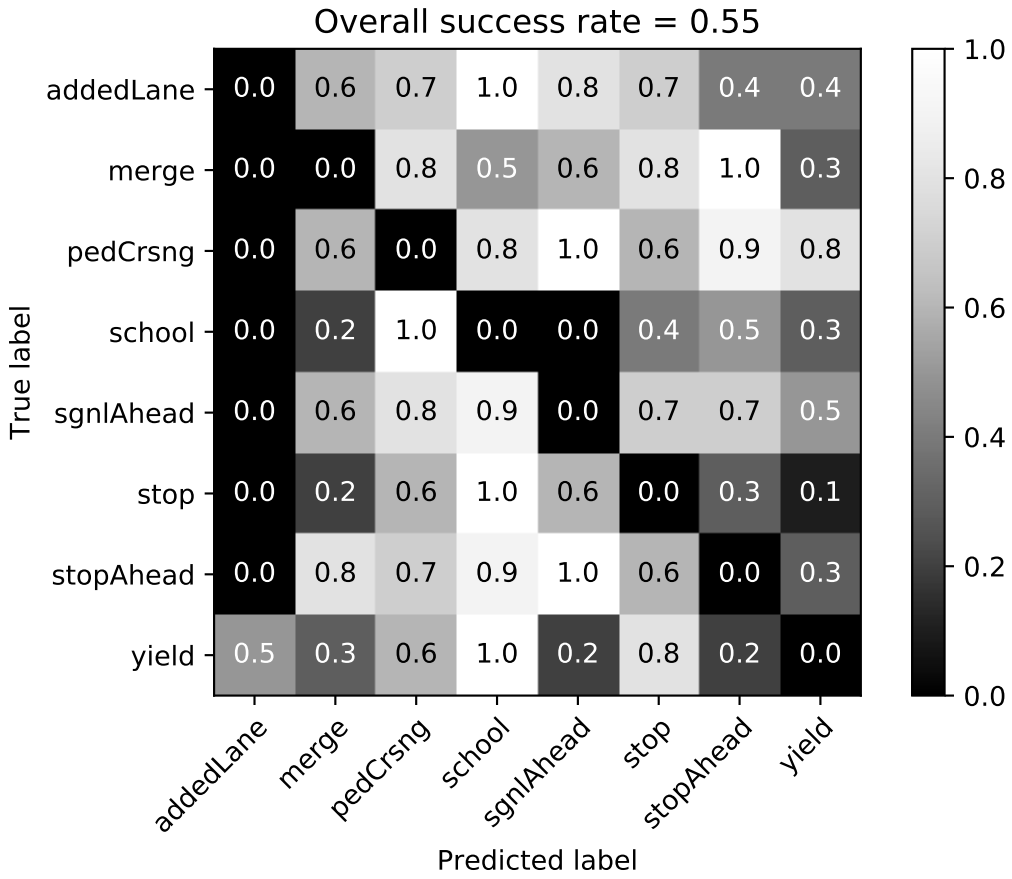


Figure 3.23: Misclassification matrix of the beamsplitting method at 4×4 resolutions

locations of signs, to make decisions, instead of only depending on real-time sensor perception (though this approach would not work for spontaneous appearance of objects, e.g., in the context of collision avoidance). Sensor redundancy/fusion could also be helpful: autonomous vehicles could be equipped with multiple cameras and/or other types of sensors, such as LiDARs and radars, which would at least increase the cost of the attack by requiring the attacker to target multiple sensors. However, a powerful attacker may be able to attack LiDARs [80], radars [26] and cameras simultaneously to defeat sensor fusion. Finally, temporal consistency via object tracking (e.g., "the object should not have appeared from nowhere of a sudden.") may also be used to detect the attack, or at least complicate it.

3.9 Chapter Summary

In this work we presented GhostImage attacks against camera-based object classifiers. Using common optical effects, viz. lens flare/ghost effects, an attacker is able to inject arbitrary adversarial patterns into camera images using a projector. To increase the efficacy of the attack, we proposed a projector-camera channel model that predicts the location of ghosts, the resolution of the patterns in ghosts, given the projector-camera arrangement, and accounts for exposure control and color calibration. GhostImage attacks also leverage adversarial examples generation techniques to find optimal attack patterns. We evaluated GhostImage attacks using three image datasets and in both indoor and outdoor environments on three cameras. Experimental results show that GhostImage attacks were able to achieve attack success rates as high as 100%, and also have potential impact on autonomous systems, such as self-driving cars and surveillance systems.

Chapter 4

A Model-based Countermeasure for Object Detection Systems against Flare-based Perception Attacks

4.1 Motivation

As GhostImage attacks are designed against camera-based object recognition systems (Fig. 3.2), potential defenses could be applied to the hardware (e.g., the camera), as well as the software (e.g., the object recognition algorithm, the sensor fusion algorithm, and/or a ghost detector). In this section, we analyze the strengths and weaknesses, as well as challenges of these common defense options so as to motivate our proposed defense, which is based on spatiotemporal consistency. Four categories of countermeasures across hardware and software are discussed.

Hardware defense: The most straightforward countermeasure to GhostImage attacks is flare elimination, possibly by using a lens hood. However, lens hoods are generally not favored as they reduce the angle of view of the camera, which is unacceptable for many autonomous vehicle and surveillance applications. Interestingly, there are so-called liquid lenses that can change focal length by reshaping themselves, which results in only one reflection path and hence fewer flares. Such lenses, however, have not been widely adopted nor are they generally available in commercial camera systems [175].

Robust ML models: A complementary line of defense would be to make neural networks (the basis of modern object detectors) robust to GhostImage attacks. Existing approaches against adversarial examples (e.g., [39, 40, 72], etc.) are ill-suited for this task, however, as these defenses were not designed for arbitrarily large perturbations [152] and GhostImage attacks need not necessarily follow the constraints placed on traditional adversarial examples, in which perturbations do not have to be bounded within a small norm.

Sensor fusion: Another complementary approach of defense is to exploit prior knowledge, such as GPS locations of signs, to make decisions, instead of only depending on real-time sensor perception (though this approach would not work for spontaneous appearance of objects, e.g., in the context of collision avoidance). Sensor redundancy/fusion could also be helpful: autonomous vehicles could be equipped with multiple cameras and/or other types of sensors, such as LiDARs and radars, which would at least increase the cost of the attack by requiring the attacker to target multiple sensors simultaneously. For example, a recent fusion-based detection scheme [176] is based on cross validation of depth estimation from three cameras. A GhostImage attacker could defeat this approach by inducing ghosts at different pixel coordinates on those three output images so that the depth estimation is still consistent across three cameras.

Ghost detection: One can pre-process the image *before* feeding it to the machine learning model. During pre-processing ghosts can be detected and then eliminated. Existing flare elimination algorithms [139, 177] are limited to spot flares that are much smaller than ours, thus are ineffective against GhostImage attacks. On the other hand, detecting large ghosts is challenging as they are typically transparent [174]. In evaluating this line of defense, we developed our own adversarial flare detection algorithm using conventional computer vision (CV) techniques, such as edge detection and shape recognition, but the detector did not generalize well as too many parameters needed to be manually updated for different inputs. Finally, using neural nets to detect ghosts [178] may enlarge the attack surface because neural nets themselves are susceptible to adversarial attacks, just as traffic sign classifiers are to system-aware GhostImage attackers. This motivates us to introduce, in Sec. 4.3, a detection algorithm that is used *after* the object recognition but before the decision-making processes (Fig. 3.2). We rely on white-box knowledge of the camera lens configuration and employ spatiotemporal consistency.

4.2 System and Threat Model

4.2.1 System Model

We assume an end-to-end camera-based object detection system in which a camera captures an image of a scene with objects of interest. The image is then fed to an neural network for object detection. The results include the class of the object, as well as its size and location, labeled as a bounding box. We assume the defender has the white-box knowledge of the lens configuration in order to predict ghost locations given light source location. Note than such an assumption can be relaxed via black-box test on the same camera system of the same model (Chapter 3.4.2). The object detector is able to detect light sources [179] such as the sun, a light bulb of a projector, etc. We assume the camera is in motion, be it installed on an autonomous vehicle of which the speed is above zero, or a home security camera with pan-tilt-zoom funtionalities [180, 181].

4.2.2 Threat Model

In an attack against the camera perception of an autonomous system, the attacker uses GhostImage techniques to spoof a spurious/incorrect object into the system, with the goal of controlling its actuation. For example, a self-driving car could be made to decelerate because it recognizes a non-existent stop sign created by the attacker. Depending on the actual design of an autonomous system, which includes the perception, planning, and control module, the adversarial object generally needs to be induced consistently for a period of time in order for the attacker to successfully induce undesired actuation.

In terms of image frames, let us assume that the attacker needs to consistently inject an adversarial object for at least k frames where $k \geq 2$ for the victim system to react as the attacker intends. For creation attacks, the attack is regarded as successful when the injected ghost has been recognized as the target class for at least k consecutive frames. For alteration attacks, an attack is regarded as successful when the injected ghost has overlapped with the target object in the image and altered the classification result, to the targeted class, for at least k consecutive frames.

4.3 Model-based Attack Detection

To defend against GhostImage attacks, we propose a spatiotemporal consistency-based algorithm to detect them. For more defense ideas, please refer to [6]. We first define the attack model in Sec. 4.2.2 (which is stronger than the one in Sec. 3.2), before introducing the general attack detection strategy (Sec. 4.3.1). We discuss how to detect attack existence, and eliminate false positives in Sec. 4.3.2 and Sec. 4.3.3, respectively. We present our security analysis considering measurement errors in Sec. 4.3.4, and evaluation results in Sec. 4.4. Finally, Sec. 4.5 presents defense discussion.

4.3.1 Detection Overview

Algorithm 1: GhostImage Attack Detection

```
1  Function IsClose(F):
2      s ← FindLightSource(F)
3      G ← CalculateGhosts(s)
4      B ← ObjectDetection(F)
5      for G ∈ G do
6          for B ∈ B do
7              if dist. between G and B is less than T then
8                  return TRUE
9      return FALSE
10 while a new frame F(t') comes do
11     F(t) ← The frame captured Δt ago
12     if IsClose (F(t)) and IsClose (F(t')) then
13         Raise an alarm
```

Our detection algorithm reduces the problem of attack detection to verifying whether there is a ghost overlapping with an object (that is detected by an object detector): If so, the algorithm raises an alarm. This is based on the fact that if an attack has, or is about to, succeed, there must be at least one frame that contains such an overlap (Sec. 4.3.2). Note that the latter is only a necessary condition of the former, but not a sufficient one. For instance, a

natural light source can produce a ghost that accidentally overlaps with the object, i.e., a false positive.

In order to eliminate such false positives, we consider two frames taken from a moving camera: If both frames contain ghost-object overlaps, the algorithm can confidently claim an attack (Sec. 4.3.3). We would like to underline that only when considering false positives do we require two frames from a moving camera. If false positives for some objects are acceptable to the system designers (out of, say, an abundance of caution to always operate safely), one frame is sufficient and, additionally, the camera can be stationary.

The benefit of our detection approach is threefold. First, it allows for attack prevention since two frames are sufficient for detection; a decision can be made before the attack succeeds if $k > 2$. Second, it detects failed attacks, too, such as an attack attempt that had manipulated only k' frames where $2 < k' < k$, or an attack where the ghost was too weak to alter the object class. Third, it requires only the result from camera-based perception; it does not rely on the information from other sensors, nor involve other modules such as the planning or control, which minimizes the attack surface.

4.3.2 Attack Existence Detection

Here, we discuss how to detect an attack assuming there is an attack, i.e., true positive detection. We find that in order to detect GhostImage attacks, the defender must detect the attack mechanism; viz., the introduction of objects through the lens flare effect, which indubitably introduces ghosts. Recall that the attacker aims to either create an image of a fake object (creation attacks), or alter an image of a real object into something else (alteration attacks). Either way, the compromised image must contain at least one (adversarial) object that is able to be detected by the object detection algorithm being used (otherwise the attack would not be considered successful, so there is nothing to defend against). Therefore, the defender only needs to pay attention to those detected objects.

Among those detected objects, the defender now needs to distinguish adversarial objects from legitimate ones. To achieve this, the defender can detect all ghosts and then check whether there is a ghost overlapping with an object: if so, then this object is considered adversarial. To detect ghosts, we assume that the defender possesses the configuration of the

lens system (white-box knowledge), which can be used to derive the pixel coordinates of all possible ghosts given the coordinates of the light source [137]. In this way, the defender avoids recognizing ghosts directly using existing CV algorithms or neural nets (which themselves are vulnerable to adversarial attacks [19]), but rather recognizes light sources, which is easier and more robust [179].

In fact, even in the case where such knowledge is not available, the defender can always follow the attacker's end-to-end, black-box-style analysis (Sec. 3.4.2) to estimate the pixel coordinates of ghosts. In any case, we may assume that the attacker is in possession of white-box knowledge of the lens system, as it does not affect the performance of our defense algorithm because *it is the physical law (instead of a secret) that reveals the attack.*

Formally, given the coordinates of the light source A, the defender calculates a set of (center) coordinates of potential ghosts $\mathcal{G}$. The object detection algorithm (e.g., YOLO) returns a set of (center) coordinates of detected objects $\mathcal{B}$. With $\mathcal{G}$ and $\mathcal{B}$, the defender raises an alarm if and only if there exists a pair of $G \in \mathcal{G}$ and $B \in \mathcal{B}$ whose Euclidean distance is less than a threshold $\mathcal{T}$. See Function `IsClose` in Alg. 1.

4.3.3 Elimination of False Positives

Unfortunately, using the above strategy alone may raise false positives. For example, a natural light source, such as the sun or streetlights, may induce a ghost that coincidentally overlaps with an object. To eliminate such false positives, we can utilize the principle of spatiotemporal consistency: Only when two frames from a moving camera yield an overlap between ghosts and objects would the detector raise an alarm (Alg. 1). A moving camera captures the dynamic of the scene which enables us to verify the spatiotemporal consistency that an attack would break.

The assumption of a moving camera is reasonable for the systems under consideration and doesn't produce a burden on their designers. For example, when autonomous vehicles are in motion so are their cameras. When vehicles are parked or otherwise stationary, we would argue that the consequence of an attack is negligible, therefore the importance of defending against an attack in such cases is low. Similarly, the cameras in surveillance systems [180,181] are usually able to pan and/or tilt themselves (e.g., to track an object). They can either keep

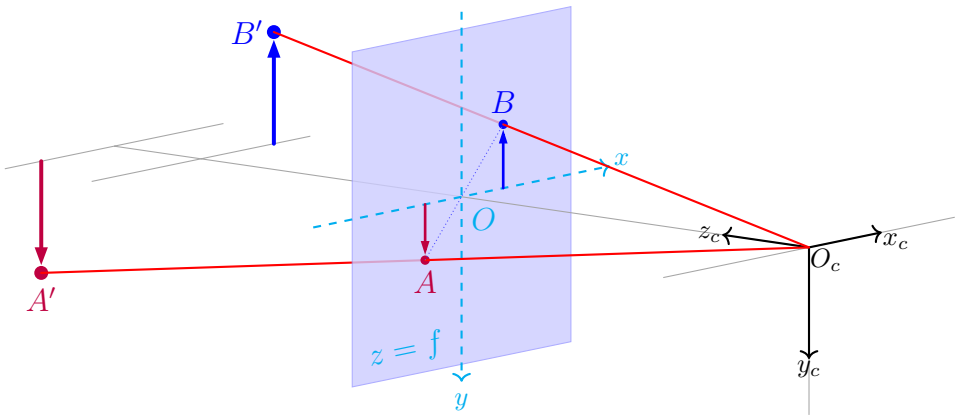
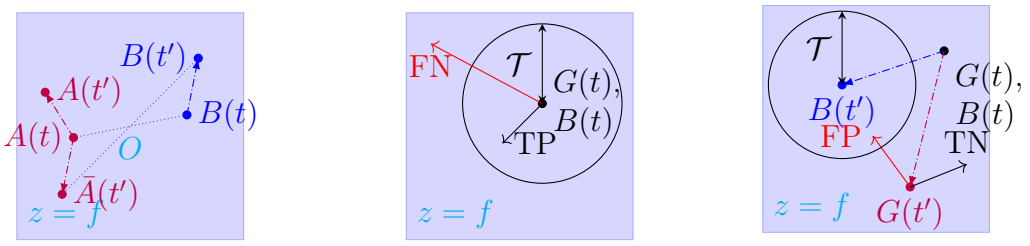


Figure 4.1: Camera Coordinate System

moving, or move themselves only when necessary (e.g., when detect a ghost-object overlay). Last but not least, we note that only when we need to eliminate this type of false positive do we require a moving camera. If the system could tolerate a slightly higher false positive rate, such an assumption might not need to be made as such false positive cases are rare.

In the rest of this section, we show that *it is impossible for a stationary, natural light source to cause a ghost that overlaps with the image of a stationary, actual object for more than one frame taken by a moving camera, when the light source and the object are at different 3-D locations.* We analyze a forward-moving vehicle to show how spatiotemporal consistency can be utilized on a moving camera to eliminate false positives caused by natural ghosts. In particular, we show that a system of three equations will eventually reveal the real-world location (3-D) of a light source, and that an object must be at least very close, due to measurement error (Sec. 4.3.4), to this location in order for the ghost to overlap with the object image consistently in more than one frame. As a natural light source is unlikely to produce two or more frames all with ghost-object overlaps, we can confidently state that a GhostImage attack is the cause of the apparent object.

Specifically, let us define the real-world coordinates of the natural light source and the natural object at time t as $A'(t) = (x_{A'(t)}, y_{A'(t)}, z_{A'(t)})^\top$ and $B'(t) = (x_{B'(t)}, y_{B'(t)}, z_{b'(t)})^\top$ respectively (Fig. 4.1). The world 3-D coordinate system is defined by (x_c, y_c, z_c) and is centered at $O_c = (0, 0, 0)$. The camera (at O_c) with focal length of f is facing towards $(0, 0, 1)$. The plane $z = f$ is the image/frame plane, which may be seen as the CMOS image sensor (the actual CMOS sensor is at $z = -f$). The origin of the image 2-D coordinate system is



(a) **(b)** **(c)**

Figure 4.2: **(a)** The frame plane at two timestamps. At time t, the images of the light source and the object was at $A(t)$ and $B(t)$ respectively. The ghost caused by $A(t)$ was also at $B(t)$ (i.e., an object-ghost overlap). Due to the movement of the vehicle (towards $(0, 0, 1)$), the object moved to $B(t')$. In order to have object-ghost overlap again in the second frame, the light source of an attacker should move to $\bar{A}(t')$, but a natural stationary light source would move to $A(t')$ instead, because of the geometry caused by the vehicle's movement. **(b)** shows TP and FN in a single time step. **(c)** shows two time steps since FP detection utilizes spatiotemporal consistency across time steps.

at its center, which is defined by (x, y). The object (e.g., a STOP sign) locates at B' in the world (3-D), and is projected to B on the image (2-D). Similarly, the light source locates at A' and A. The line AB crosses O. We assume both are stationary in the real-world. The corresponding pixel coordinates of them in the camera-captured image are denoted as $A(t) = (x_{A(t)}, y_{A(t)})^\top$ and $B(t) = (x_{B(t)}, y_{B(t)})^\top$ respectively. The image center is at O. Given the camera matrix M_c, at time t, we can derive the pixel coordinates of $A(t)$ based on its real-world coordinates $A'(t)$ by the well-known homogeneous transformation (Eq. 3.5). Similarly, we can derive $B(t)$, as well as $A(t')$ and $B(t')$ at time t' where $t' > t$ (Fig. 4.2a). We define $\Delta t = t' - t$ (Alg. 1), which will be discussed in Sec. 4.5.

To simplify the analysis, we assume that the camera orientation is $(0, 0, 1)^\top$ for all time. In addition, because the camera is always the center of the coordinate system (Fig. 4.1), as the vehicle is moving forward in the real-world, the object and the light source are both moving backward by the same distance. Let us define the camara displacement as $D = (\Delta x, \Delta y, \Delta z)^\top$ where $\Delta y = 0$ (altitude) for simplicity. With that, we have

$$A'(t') = A'(t) + D, \quad B'(t') = B'(t) + D. \tag{4.1}$$

For a false positive to occur the ghost G caused by the light source A overlaps with the image of the object B, i.e., $B = G$ (otherwise it is not a positive detection). At time t, the image of the light source $A(t) = (x_{A(t)}, y_{A(t)})^\top$, the image of the object $B(t) = (x_{B(t)} y_{B(t)})^\top$ and the

frame center $O = (0, 0)^\top$ must be on the same straight line (Fig. 4.2a). The same principle applies to time t' as well. Thus, we have our first two equations in (4.2).

$$\frac{x_{A(t)}}{y_{A(t)}} = \frac{x_{B(t)}}{y_{B(t)}}, \quad \frac{x_{A(t')}}{y_{A(t')}} = \frac{x_{B(t')}}{y_{B(t')}}, \quad \frac{x_{A(t')}}{x_{B(t')}} = \frac{x_{A(t)}}{x_{B(t)}}. \tag{4.2}$$

In addition, recall that the source-ghost ratio $r = \overline{OB}/\overline{OA}$ must be identical at any time (Sec. 3.4.2). Thus, for the case of two frames we arrive at the third equation in (4.2).

From a high level, Eqs. 4.2 capture the optical principle of the lens flare effects and overlaps, while Eqs. 4.1 capture the moving camera assumption. The homogeneous transformation captures the optical imaging principle. Together these produce the conditions under which spatiotemporal consistency exists. That is,

$$\frac{y_{A(t')}}{y_{B(t')}} = \frac{x_{A(t')}}{x_{B(t')}} = \frac{x_{A(t)}}{x_{B(t)}} = \frac{y_{A(t)}}{y_{B(t)}}.$$

When we plug all of them back to the homogeneous transformation (which derives $A(t)$ from $A'(t)$, and $B(t)$ from $B'(t)$), we arrive at $A'(t) = B'(t)$, which means the light source has to be placed at the same real-world location as the object, or at least very close. The chance that there is a naturally occurring source-ghost pair with a ratio of one is rare, not to mention in such a case that the ghost would be overwhelmed by the image of the light source itself (burst effect).

In conclusion, a natural light source is unlikely to cause two or more frames all with ghost-object overlaps. Therefore, if two or more frames indicate such an overlap, the defender can confidently claim a detection of GhostImage attacks with a low chance of false positives.

4.3.4 Security Analysis

Because there will be errors in determining the pixel coordinates of objects, light sources, ghosts, etc., we evaluate the True Positive Rate (TPR) and False Positive Rate (FPR) under different thresholds $\mathcal{T}$. We again use the forward-moving vehicle scenario to study the impact of different camera displacements, and different real-world configurations of objects and light sources.

First of all, we estimate the error distribution using the same data that we collected when we moved around the flashlight in front of the camera (See Sec. 3.4.2), for which we run a light source detection algorithm that is based on edge detection and shape recognition [179]. As we do not have detailed information on the lens configuration, we estimate the ghost locations using the source-ghost ratio (Eq. 3.6) approximated in Sec. 3.4.2 for our setup. Results show that the error follows a bivariate Gaussian distribution, i.e., $\epsilon \sim \mathcal{N}(\boldsymbol{\mu}_\epsilon, \boldsymbol{\Sigma}_\epsilon)$, where

$$\boldsymbol{\mu}_\epsilon = \begin{bmatrix} \mu_\epsilon \\ \mu_\epsilon \end{bmatrix}, \quad \boldsymbol{\Sigma}_\epsilon = \begin{bmatrix} \sigma_\epsilon^2 & 0 \\ 0 & \sigma_\epsilon^2 \end{bmatrix}.$$

For our experimental setup, we have $\mu_\epsilon = 0.014$, $\sigma_\epsilon^2 = 0.004$ with the image size normalized to one.

True Positives (TP) and False Positives (FP) due to the measurement errors (marked in solid single arrows). The blue square plane $z = f$ is the (imaginary) CMOS image sensor. In Fig. 4.2b, we consider only one time stamp because TP depends only on the measurement error, but in Fig. 4.2c for FP, we need to consider two timestamps as FP also relies on spatiotemporal consistency.

For true positives (Fig. 4.2b), we first derive the TPR r_{TP} for a single frame; the TPR for two frames, assuming the error distribution is i.i.d. across frames, is then r_{TP}^2. For simplicity, we analyze the worst case scenario where the attacker is able to track the camera perfectly, which means the center of the adversarial ghost G is always at the center of the object image B, i.e., $B = G$. Assuming the ground truth is both $G(t)$ and $B(t)$ being at the same pixel location (because the attacker is able to track the camera, i.e., the worst case for the defended system), In our analysis we note that the measured pixel location of ghosts may be incorrect (Fig. 4.2b). On one hand, the estimated ghost may fall out of the circle $G(t)$ with Radius $\mathcal{T}$, causing a false positive (the red arrow); on the other hand, the error may be small and the ghost is still within the circle, thus a true positive. The derivation of r_{TP} is the integral of the probability density function (PDF) of the error ϵ over the circle that centers at $B(t)$ with Radius $\mathcal{T}$ (Fig. 4.2b). For TP, the integral is a special case of the offset circle probability [182] with zero offset.

The worst case scenario of FPRs is different: A natural ghost $G(t)$ and an object $B(t)$

located at the same pixel coordinates in the first frame and in the next frame they move to $G(t')$ and $B(t')$, respectively, due to the displacement of the camera (Fig. 4.2c). A larger error may result in a false positive (falling into Circle $B(t')$), while a smaller error may produce a true negative. The calculation of FPR is different than TPR, as the TPR's derivation is based solely on the error distribution because the attacker's capability of tracking the camera compensates for the impact of the 3-D location of the adversarial light source A', and the camera displacement D. For the worst-case FPR, we need to consider different combinations of the natural light source locations and camera displacements because their variation produces different distances between $G(t')$ and $B(t')$, denoted as $d_{GB} = \overline{G(t')B(t')}$, that further produces differing FPRs. We use offset circle probability [182] to calculate the FPR r_{FP}, which is the circular integral of the error's PDF, where the offset circle centers at $B(t')$ with Radius $\mathcal{T}$ (Fig. 4.2c). For FP, the error is zero at $G(t')$ hence the offset is d_{GB} of the size of the common area of Circle $B(t')$ with Radius $\mathcal{T}$ and Circle $G(t')$ with Radius, divided by the area of the latter circle (because ϵ is uniformly distributed).

4.4 Evaluation

4.4.1 Varying Object and Ego Displacement

Setup: We ran a numerical evaluation with settings described as follows. For simplicity, we assume a pinhole camera (Fig. 4.1) with a camera matrix

$$M_c = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where $f = 30$ mm is the focal length, and $p_x = p_y = 0$ which means the camera origin is at the image origin. The size of the CMOS sensor is 7×6 mm. At time t, let us suppose the object is at $B' = (1\,\text{m}, 2\,\text{m}, 20\,\text{m})^\top$ and we iterate the natural light source's real-world location A' around B' to show the impact of configuration on detection performance. Because we are interested in the worst-case FPR, for each instance of A', we assign the source-ghost ratio r with a value that would result in perfect ghost-object overlays at the first frame, i.e.,

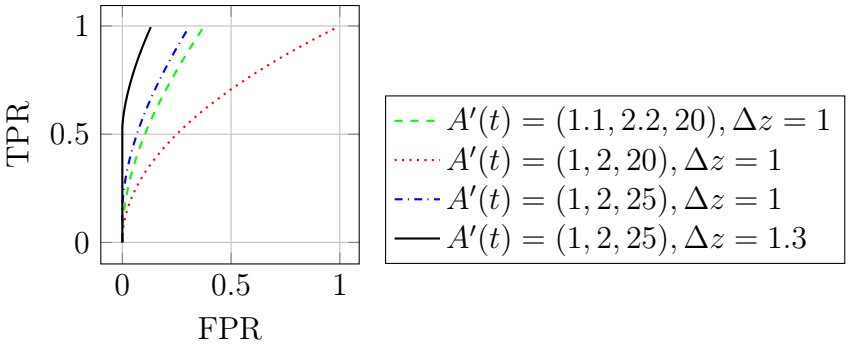


Figure 4.3: Receiver Operating Characteristic (ROC) curves for varying camera displacements $(\Delta z/2, 0, \Delta z)^\top$ and varying real-world locations of the natural light source, A'. The object location is fixed at $B'(t) = (1, 2, 20)^\top$.

$r = \overline{OB(t)}/\overline{OA(t)}$.

Results: We present the distribution of equal error rates (EERs) in terms of different real-world locations of the natural light source in Fig. 4.4a. There are only two degrees of freedom, $x_{A'(t)}$ and $z_{A'(t)}$, that decide the 3-D location of the natural light source because the it needs to be on the plane, OBB' (Fig. 4.1), otherwise there would not be a ghost-object overlap in the first frame. We take the object to be located at the center of the plot, i.e., $B'(t)$. We set $\Delta z = 3\,\text{m}$ and $\Delta x = 0.75\,\text{m}$. We iterate $x_{A'(t)}$ from $0\,\text{m}$ to $2\,\text{m}$, and $z_{A'(t)}$ from $10\,\text{m}$ to $30\,\text{m}$. We see that a higher EER occurs when A' is close to B', but it decreases to zeros as A' is placed away from B'. This supports the theoretical conclusion made in Sec. 4.3.3 that in order to have ghost-object overlaps in two frames, the natural light source needs to be placed at or close to the object in the real world.

To illustrate the impact of the camera movement, we test different camera displacements by varying Δx and Δz (Fig. 4.4b). The natural light source is at $A'(t) = (1.25, 2.5, 22)^\top$. We make three observations: First, larger Δz (e.g., higher car speeds, or lower sampling rates) yields a lower (better) EER. This is because when the vehicle is moving faster (i.e., Δz is larger), the ghost is further away from the object image in the second frame (i.e., d_{GB} is larger). Therefore the integral over Circle $B(t')$ of the error distribution becomes smaller (Fig. 4.2c), which results in a lower false positive rate. Meanwhile the true positive rate remains constant because it depends only on an error distribution that is i.i.d. Second, driving slightly towards the left or the right results in a lower EER because such movements change the real-world relative, geometrical configuration (of the camera, the light source, and

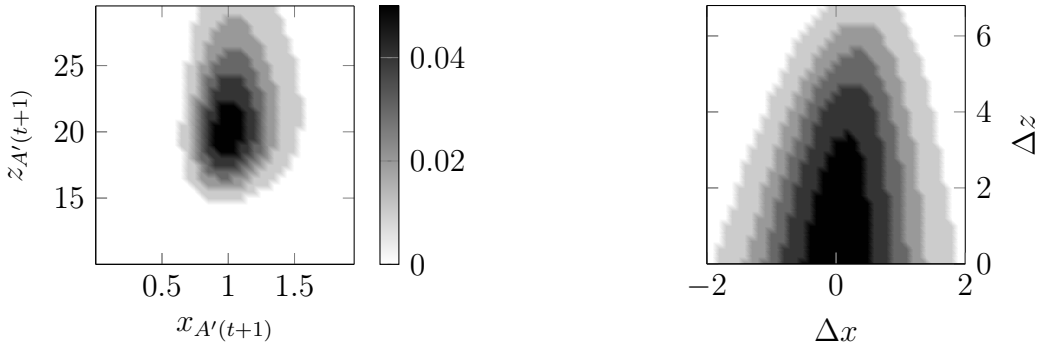


Figure 4.4: Equal error rate (EER) distribution under (a) varying real-world locations of the natural light source, with a fixed camera displacement $(0.75, 0, 3)^\top$, and (b) varying displacements of the camera, with a fixed natural light source location $A'(t) = (1.25, 2.5, 22)^\top$. Both figures share the same color bar. For both figures, the object is at $B'(t) = (1, 2, 20)^\top$, and the unit of all axes is meter.

the object) more significantly than moving only forward. Third, movement towards to the left yields a lower EER than to the right because of the configuration as well. To explain this we note that if $A'(t) = (1.25, 2.5, 18)^\top$ or $A'(t) = (0.9, 1.8, 22)^\top$ (i.e., the natural light source is closer to the camera than the object along either the x or z axis), the inclination of EER will be on the other side.

4.4.2 Moving Natural Light Sources

Here, we relax the assumption of stationary, natural light sources and use a real-world dataset to test whether a moving natural light source such as the head light of an incoming vehicle can produce false positives.

Setup: From the BDD100K dataset [183], we randomly select the 100 traces of the headlights and 100 traces of traffic signs and the average length of these traces is three seconds at 5 Hz (15 frames per trace). For each pair of a headlight trace and a sign trace, we aim to test the ghost created by the headlight is close enough (determined by the threshold $\mathcal{T}$) to the sign for two or more consecutive frames (which is our detection criterion, otherwise it is not a false positive). Because they may start to appear at different time, we shift one trace by various steps. Once a combination of two traces, $G(t), 1 \leq t \leq M$ and $B(t), 1 \leq t \leq N$, and the shift step k has been decided, we compute the Euclidean distances between $G(t)$ and

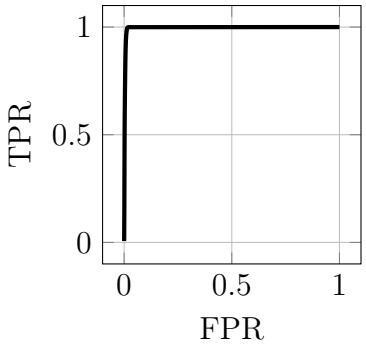


Figure 4.5: The ROC curve of moving headlights.

$B(t+k)$, and between $G(t+1)$ and $B(t+k+1)$: If both distances are less than $\mathcal{T}$, we mark this combination as a false positive. Therefore, given a threshold $\mathcal{T}$, we can calculate the false positive rate. The true positive rate calculation is the same as in Section 4.3.4.

Results: We plot the ROC curve in Figure 4.5. We can see that the moving headlights rarely cause false positive rates. The reason is that the dynamics of the ghosts created by the headlight of an incoming vehicle is significantly different than the dynamics of a traffic sign. They may overlap for one frame, but continuous overlapping for more than two frames is rare as their directions are different. See Figure 4.2a for an illustration.

4.5 Discussion

The results in Figs. 4.4a and 4.4b are based on a simple light source detection algorithm [179] and an approximated source-ghost model (Sec. 3.4.2); if the defender uses a more sophisticated light source detection algorithm with higher accuracy [184], and/or a more precise source-ghost model derived from white-box lens configuration, an even lower EER can be achieved.

Recall that $\Delta t = t' - t$ is the time duration between two moments when the attack detection algorithm samples two images. The higher the sampling rate ($1/\Delta t$), the quicker a detection decision can be made; therefore, high sampling rates may seem preferable. However, when the vehicle is slow a high sample rate results in a small camera displacement, which means a higher EER. As such, the sampling rate should be proportional to the speed of the vehicle (i.e., $\Delta t \propto 1/v$) because a high sampling rates at high vehicle speeds still gives large enough camera displacements (thus lower EERs), and more importantly, it is more critical to detect attacks quickly in high speed scenarios for applications such as autonomous vehicles.

An attacker may combine GhostImage attacks with adversarial patches against object detection [152] where the label of the target object can be altered by adding a patch that does not need to overlay with the object. Such attacks can be detected and mitigated, by removing the patch, and then checking whether the object label is changed or not: If so, then the patch is considered adversarial, and the ground-truth label is the one after removal. The defender can locate the patch because it was conveyed via ghosts whose locations can be estimated (See Sec. 4.3.2).

Similarly, the threshold $\mathcal{T}$ should also be decided based on the vehicle speed, as well as which of the true positive rate or false positive rate is more important to the application. Recall that $\mathcal{T}$ is the threshold that is used to determine whether a ghost-object pair is close to each other or not: If yes for two frames, an attack is detected. On one hand, if a small threshold is used, the true positive rate may be low because the object detector may return bounding boxes with slight errors. On the other hand, if a large threshold is used, many false positives may arise. As a result, a dynamic threshold is suggested. For example, in the case where the vehicle is moving forward,

$$\mathcal{T} \propto \Delta z, \quad \text{with } \Delta z = \frac{v}{s},$$

where Δz is the distance that the vehicle moves between two examined frames (Sec. 4.3.3), v is the speed of the vehicle, and s is the number of frames per second that the attack detection algorithm takes. In an extreme case where the car stops and coincidently there is a natural ghost overlapping with a STOP sign, which results in more than two frames with ghost-object overlaps, $\mathcal{T}$ should be zero so no detection would be made.

Algorithm 1 works for both creation and alteration attacks, because as long as there is a detected object (either created or altered) that has the similar pixel location to a ghost, we claim it is an attack unless we eliminate it as a false positive. This applies to both stationary and moving attackers. In any case, we may utilize the data-driven approach to verify the spatiotemporal dynamics of the object with its classification result (Chapter 5).

4.6 Chapter Summary

We proposed a model-based attack detection algorithm to detect adversarial ghost attacks. The detection algorithm is based on the idea of spatiotemporal consistency and reduces the problem of detecting ghost-based attacks into the problem of verifying if there is an overlapping between a ghost and detected object. Evaluations with real-world data shows that our detection algorithms produces a worst-case equal error rate of only 5%.

Chapter 5

Object Misclassification Attack Detection Using Data-Driven Spatiotemporal Consistency Verification

5.1 Introduction

Object detection and tracking (ODT) is an essential component in the perception module of an autonomous system, such as self-driving cars [15–17], robots [185], drones [128], etc., as they need to perceive the environment and understand the dynamics of the surrounding objects (what/where they are, and where they are going) so that actions can be taken accordingly. In Baidu Apollo [15], for example, cameras are used to capture video frames that are fed to machine learning (ML) algorithms for ODT [90]; the results of which are then used for decision-making, e.g. to stop the car for a pedestrian who is, or is about to, cross the street.

Recent work has explored attacks against perception approaches/pipelines by physically modifying objects [5, 22, 82, 83, 186] or by manipulating the sensing mechanisms (e.g., noise produced by (in)visible light [6, 8, 31, 61], acoustic signals [9], radio-frequency electromagnetic waves [132, 187] etc.) such that the subsequent machine learning algorithms using the output of the sensor produces results that are intended by the attacker. The goal of an attacker in launching these *perception attacks* against object detection algorithms can be generally categorized into three types: object disappearance [82], fake object creation [22], and misclassification of detected objects [5]. As a consequence of a particular false perception result, inappropriate decisions and actions may be taken by the planning module of an autonomous system that could jeopardize safety; e.g., a car stops on a highway because it recognizes the car in front of it as a person.

Among those aforementioned attacks, this work is focused on misclassification attacks

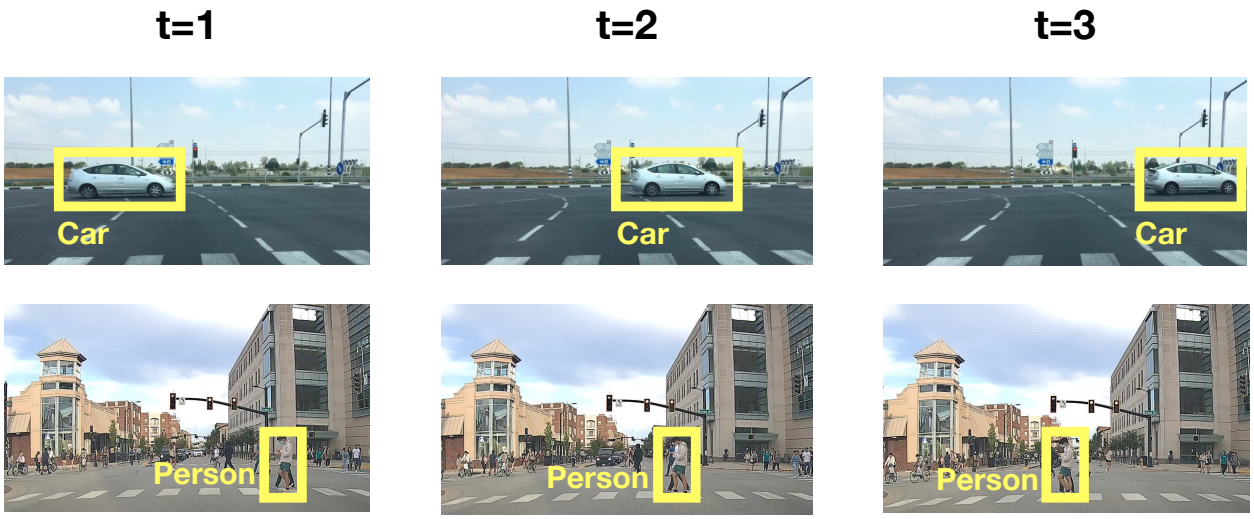


Figure 5.1: Typically, a person moves slower than a car. Also, their bounding boxes are more vertical than a car's.

against vision-based object detection and tracking for autonomous systems. In particular, we consider an attacker who has white-box knowledge of the system, and aims to change the classification result of the target object by inducing adversarial noise to video frames. For example, an adversary may place an LCD monitor on the back of a vehicle and display adversarial noise so that the following vehicle recognizes it (the predecessor) as a person [4].

Most existing defenses against these attacks are either specific to the sensing modality (e.g., LiDAR [31], GPS [32], IMU [33]) or, for those that do consider vision, often assume threat models that are specific to one particular attack methodology, e.g., norm-bounded attacks [34] or adversarial patch attacks [35]. The root cause of such specificity is that these defenses focus on the sensing *input* to the ML models, therefore once a new type of attack appears, a new defense is required.

Instead of focusing on the input, in this work we aim to develop a more general defense approach by validating the *output* of the ODT algorithms in order to detect misclassification attacks. For each detected object the ODT system outputs predictions for both the class and trajectory (represented by a *tracker* in the form of a sequence of bounding boxes (bboxes) over time). Our main idea is to verify the spatiotemporal components of the tracker by cross-checking with the class prediction; i.e., the perceived spatial and temporal attributes of different object are unique and describe a type of signature that can be verified over time.

For example, for an object to be classified as a person, intuitively it has to not only look like a person but also move like a person. In other words, there is an inherent connection between classes and trackers but unfortunately existing ODT systems predict them separately. Our approach, on the other hand, fuses both together to verify the consistency between them; when the consistency is lacking an attack is indicated.

As a motivating example, we consider a self-driving car that uses a video-based vision mechanism [16] (Fig. 5.1). On one hand, when considering a single frame, the shape and location (spatial property) of an object class can be fingerprinted. For instance, the bbox of a person is usually more vertical thin than of a car, and it typically appears on the edges of the frame (on the sidewalks) rather than at the center when the ego-vehicle is driving. On the other hand, when considering multiple frames overtime, the temporal property of bbox dynamics is distinctive among different object classes. For example, a car driving across an intersection typically moves faster than a pedestrian does.

Although such empirical knowledge seems intuitive to humans, in order to build an effective defense, several key challenges/research questions need to be resolved. First, we must determine which spatiotemporal attributes/features of an object to use so that they are statistically-sufficient to distinguish different object classes. Second, we need to effectively and efficiently learn the spatiotemporal property from these features in order to produce low false positive and negative rates for attack detection. Lastly, the defense needs to be evaluated against practically realizable attacks, under real-world scenarios, assuming a strong threat model with adaptive attacks.

We propose PercepGuard to detect perception-based misclassification attacks. In PercepGuard a recurrent neural network (RNN) is used as a sequence classifier to classify a tracker into an object class. During training the RNN learns the spatiotemporal property of bbox sequence for each object class. Our detection criterion is that, if the classification result of PercepGuard does not match that of the ODT system, an alarm will be raised (Fig. 5.2). Our evaluation with BDD100K [183], a real-world driving dataset, shows a maximum false positive rate (FPR) of only 5%, and true positive rates (TPR) as high as 99% on adversarial patch attacks [94], a prevalent attack where the patches are generated by solving an optimization problem and can be realized via printed stickers [5], TV monitors [4], or projectors [22].

However, for adaptive attackers who also aim to evade PercepGuard, our TPR drops to 85.6%. This is because when the adaptive attacker places our detection criterion as a constraint into their optimization problem, the optimizer is able to find those patches that accomplish not only misclassification but also detection evasion. E.g., to alter a car into a person, the attacker additionally changes the bboxes from horizontal to vertical for the RNN to classify them as a person. To improve the adversarial robustness, we consider contextual information such as the perceiving (ego) vehicle's velocity, its relative velocity to objects, etc., which are generally available from on-board sensors such as speed sensors and LiDARs. Specifically, we augment the feature space of the RNN so that it learns the relation among these features during training. As a result, for the example above, the RNN still classifies those altered bboxes as a car even though they are vertical, because a person should not move as fast as the ego vehicle (i.e., with a low relative velocity) when the latter moves at a high velocity. Evaluation using a Carla-simulated dataset [188] shows that with only these two contexts, the TPR increases to 99% from 85.6%. Baseline comparison with NIC [46] and SCEME [10] shows the superiority of PercepGuard as it takes advantage of context and spatiality over multiple frames. In real-world experiments, we successfully demonstrate adversarial patch attacks (using an LCD monitor or a projector on a moving vehicle), but fortunately PercepGuard is able to detect 43 out of 45 attack instances, including adaptive attacks.

5.2 Background and Related Work

In this section, we introduce the background of object detection and tracking systems and their vulnerabilities.

5.2.1 Object Detection and Object Tracking

To perceive the environment, an autonomous system needs to locate the surrounding objects, classify them, and finally track their trajectories. Modern object detection is typically achieved via a two-stage NN (e.g., Faster-RCNN [189]) that comprised of a region proposal network (RPN) and a detection network, or one-stage (e.g., YOLOv3 [1]) that achieves both as the

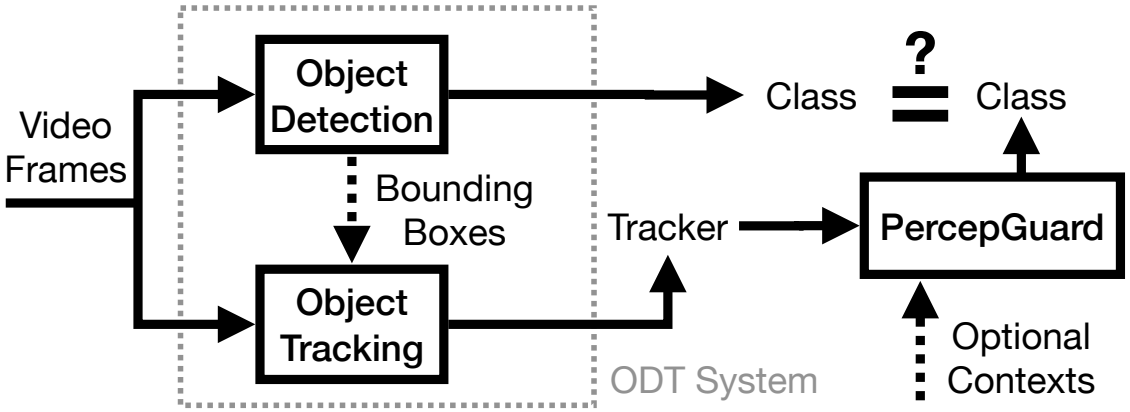


Figure 5.2: High level idea of PercepGuard

same time. For object tracking, two-stage algorithms rely on an object detector for bounding box predictions of each frame, then they associate identical objects across frames based on their momentum [190–192], while a one-stage algorithm utilizes one single NN [193] to achieve both tasks. Both types of methods eventually output a *tracker* for each object's trajectory, i.e., a sequence of bounding boxes. PercepGuard aims to analyze the tracker (optionally with contexts) to verify the object's class prediction (Fig. 5.2) that may have been tampered by an adversary. As a proof-of-concept, we evaluate PercepGuard using vision-based, two-stage systems as they are the most common and developed systems [14, 15, 90].

An attacker may create a non-existent object by exploiting a variety of modalities (camera, LiDAR, radar, etc). For example, attacks against camera sensors include injecting noise directly into the camera sensor (e.g., using visible light projection [6, 7], radio-frequency electromagnetic signals [187], etc), and exploitation of imperfections of camera sensors (e.g.,the ghost/flare effect [6, 7] and rolling shutter effect [8]), or projecting an image on environmental surfaces (e.g., sidewalks or walls [22]). However, our combination of detection heuristics and imposition of spatiaotemporal consistency over time makes these attacks unlikely to succeed as most of them either project static objects or can only maintain consistency for very few number of frames [22]. Additionally, many of the attacks are opportunistic [8]) in that they do not target consecutive frames and would thereby be detected as inconsistent.

PercepGuard detects misclassification attacks against object detection and tracking systems by verifying the spatial and temporal consistency of an object's shape and movement overtime from one single camera, i.e., a homogeneous setting. To improve adversarial robustness

against adaptive attacks, we extend PercepGuard to consider additional context information which can be obtained from multiple, heterogeneous sensors including cameras, speed sensors, or LiDAR/radar. Because PercepGuard focuses on the goal/outcome of the attack, it is agnostic to specific attack modalities/methodologies.

Existing context-based inconsistency detection methods either require multiple objects [10, 95], or focus on a different application such as image classification [38] or person re-identification [98]. Other more related works are limited to either specific sensor types, or specific noise injection methods. For example, LiDAR attacks [80] can be detected through occluding relationships between objects [31] but such relationships are only available from LiDAR systems. Similar limitation also applies to those countermeasures for GPS and IMU [33] systems etc. For those methods that defend vision-based object detection systems, they are limited to their targeted attack methods as well. Most defenses against digital domain adversarial examples assume norm-bounded perturbation (e.g., [34, 40, 46]) which cannot apply to physical/perceptional attacks where typically the noise are not bounded by norm otherwise they would be nearly impossible to realize. Those that do not have such an assumption limit themselves to a particular attack methodology, e.g., adversarial patch attacks [35,92], physically-realizable attacks [93], or DoS attacks against depth estimation [37].

NIC NIC is proposed to detect adversarial examples against image classifiers by checking the network invariant. It produces a binary detection for each individual video frame. Specifically, when an image is fed to an image classifier, NIC obtains the activation values of each individual layer (value invariants, V_i), and every two consecutive layers (provenance invariants, $P_{i,j}$), where i and j are the layer indices. They first use PCA to reduce the dimension of these invariants and obtain V_i' and $P_{i,j}'$. For each V_i', an SVM is used to produce a score v_i. For each $P_{i,j}'$, they first feed it to a fully connected layer that has M neurons where M is the number of classes, and then feed the M activation values to an SVM for a score $p_{i,j}$. Finally, an SVM is used to classify all v_i and $p_{i,j}$ into a binary decision on if the input image is adversarial or not.

The drawbacks of NIC include (1) It focuses on individual frames thus cannot utilize temporal information that we have demonstrated to be useful in detecting misclassification

attack against object detection. (2) NIC is not agnostic to attack methodologies since it assumes the attacker uses optimization-based adversarial machine learning algorithms to generate the perturbation noise, an assumption that we do not make in our threat model so that PercepGuard is able to additionally detect real-image attacks that NIC cannot detect. (3) NIC's computation overhead in both time and space is high because for each invariant there is an SVM, which makes it unsuitable for autonomous driving where the system's responsiveness (delay) is critical and the computational power onboard is limited. (3) NIC is designed for image classification which is not as widely used in autonomous driving as object detection that PercepGuard is designed for. Moreover, it is nontrivial to adapt NIC for object detection due to the architectural difference. In particular it is the M-neuron fully-connected layer that is not compatible with object detection, be they one or two-stage.

SCEME Similar to NIC, SCEME also works on individual frames but it produces a soft decision score for each object in the frame by verifying the contextual consistency among objects, backgrounds and scenes. For each region proposal in a frame, SCEME employs context-aware Faster-RCNN [SIN] to extract a context profile $u = [r, \gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$, and then feed it to an autoencoder for reconstruction loss: if the loss is above a certain threshold, SCEME claims the object (if the region contains an object) as adversarial. There is one autoencoder for each object class. To train them, SCEME uses context profiles that are collected from benign images so that they learn the distribution of a benign object's profile for a given class, thus at test time it will be able to identifies objects that are misclassified, for instance.

The drawbacks of SCEME are three-fold. Firstly, it is only compatible with two-stage object detectors that produce region features which means it does not work with one-stage algorithms such as SSD or YOLO that is more commonly used in autonomous driving due its superior detection speed and accuracy [15]. Secondly, it subjects to the richness of contexts that varies overtime. For example, there is insufficient evidence for context verification if there are only few regions being proposed. Lastly but most importantly, it only considers spatial and contextual features at one frame but ignores the temporal features across multiple frames.

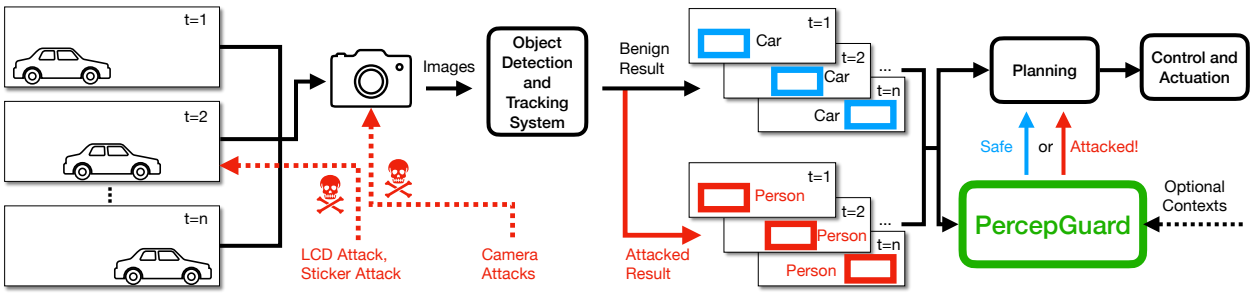


Figure 5.3: System and Threat Model. In the perception module of an autonomous system, vision sensors (e.g., cameras) perceive the environment and convert into video frames for object detection and tracking, which labels an object with class and bounding boxes. The planning module takes these labels (among other inputs) for decision-making, and finally the control module executes these decisions. When the perception module is compromised, either by physical attacks (e.g., using an LCD monitor [4], or a sticker [5]), or camera sensor attacks (e.g., [6–9]), the perception module outputs falsified labels which may mislead the planning and control modules. PercepGuard verifies these labels and alarm the system when an attack is detected.

5.3 System and Threat Model

We discuss our system model and threat model in this section.

5.3.1 System Model

We consider an object detection and tracking system (Fig. 5.3) for autonomous vehicle applications, that are equipped with vision-based perception modules. The vision input is mainly provided by images/videos captured by an on-board camera as the main sensor. We assume that the on-vehicle camera is placed at a fixed location (e.g., dashboard) with a fixed orientation as is typical for existing systems [16, 17, 194]. The images are fed to a NN for object detection [1]. The object detection result of each frame will be integrated for object tracking. Other *optional* sensors may also be included to provide auxiliary input, such as lidar/radar, and vehicle speed sensors (VSS). Their corresponding sensory data processing algorithms can also be part of the system. The planning and actuating modules are out of the scope of this work, neither are sensor fusion algorithms [32, 33].

5.3.2 Threat Model

We consider perception attacks where the adversary's objective is to alter an existing object's classification result into a target class, thus jeopardizing the behavior of the autonomous

vehicle, e.g., forcing it to stop on a highway because it recognizes the front car as a person, or vise versa. We assume the attacker is able to cause misclassification for multiple frames consecutively in order to affect the decision-making of the autonomous system [90]. The attacks can be realized via physical modification to the objects [4, 22] e.g., adversarial patches [94] which we will use for demonstration, or tampering sensors themselves [6, 9]. Therefore, the attacker must consider the *physical constraints* such as the limited patch size and magnitude to make it realizable in the real world. Instead of bounding the noise norm to make the patches imperceptible, they minimize the patch magnitude to reduce attack cost. We do not consider digital domain attacks that alter the sensory data digitally[1], since it requires access to the internal network (e.g., the CAN bus) which can potentially take over the whole system [44].

We assume the adversary possesses white-box knowledge of the sensory data processing algorithms, e.g., (hyper)parameters of the NNs, as well as the raw video frames taken by the camera. We consider two types of adversaries:

- *Detection-unaware* attackers, who is not aware of our defense and only tries to cause misclassification. To do that, they can either inject adversarial noise opportunistically, or solve an optimization problem for the optimal perturbation (to the benign video frames), in which they aim at minimizing the magnitude of the perturbation subject to succeeding in targeted misclassification.

- *Detection-aware* attackers, a.k.a. adaptive attackers, who know the existence of our defense, possesses the parameters of it, and tries to evade the detection. To do so, they additionally add a constraint of evading our detection to the optimization problem.

When more than one sensor is used, we assume the adversary can obtain and modify the sensory data.

Finally, we assume the autonomous system (including PercepGuard) runs on trusted/tamper-proof hardware, firmware and software (i.e., a trusted computing base [195]); that is, we do not consider internal threats that directly hack the CAN bus [116] or the operating system of a vehicle, as they belong to the study of intrusion detection systems [44].

[1]Although PercepGuard can also detect norm-bounded adversarial examples that span multiple video frames in the digital domain [19, 20]

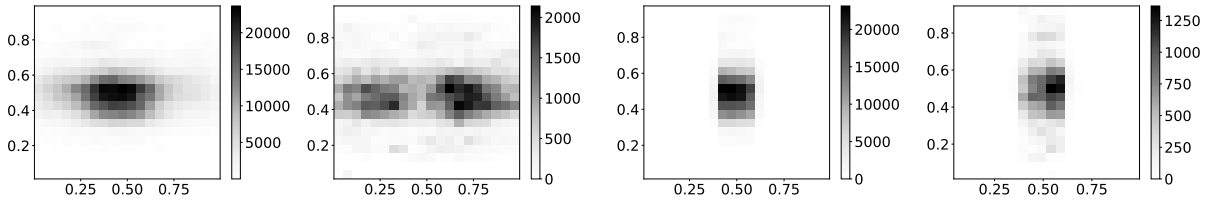
5.4 Our Attack Detection

In this section, we first state the problem of attack detection, and then give an overview of our approach with main challenges identified. Later, we introduce the basic method using bounding boxes only, and then the context-augmented detection which enhances the adversarial robustness.

5.4.1 Problem Statement and Challenges

Problem Statement: Given a series of consecutive frames captured by the vision sensor (e.g., camera), for each detected object, the object detetion and tracking (ODT) system returns the class prediction of the object c, as well as a tracker which is a sequence of bounding boxes $B = \{b_i : 1 \leq i \leq N\}$, where b_i is the bounding box at time i, and N is the number of frames in which this object is detected. A 2-D bounding box b is described by a vector $(x, y, h, w)^\top$, representing its center coordinates (x, y), height h and width w. Given (B, c), our goal is to verify whether $c = c_o$ where c_o is the ground-truth class. To achieve the detection goal, we propose to classify B into a category c' that shares the same domain of c (i.e., c' can take any value that c can). If $c \neq c'$, we regard the video frames as adversarial input (Sec. 5.4.3). Moreover, we consider stronger attackers who have white-box knowledge of our detection algorithm and try to evade it. They manipulate the frames for both B and c to make $c = c'$. We propose to incorporate additional context information to enhance the adversarial robustness of our detector (Sec. 5.4.4).

Technical Challenges: There are several challenges involved. First, the existence of such a classifier is unknown. In other words, does B provide sufficient information at all to distinguish one object class from another? Second, if such a classifier exists, it is non-trivial to design it in a way that it achieves high-accuracy, and can be trained, tested, and extended efficiently. Third, it is equally important for an attack detector to ensure low FPRs as to ensure high TPRs. Finally, it is challenging to enhance the adversarial robustness against strong attacks, such as adaptive attackers who are aware of the defense and try to evade it, and are able to compromise multiple, even heterogeneous sensors simultaneously.



(a) Car at one frame **(b)** Person at one **(c)** Car between two **(d)** Person two

Figure 5.4: Spatial and temporal histograms of bounding box center coordinates, i.e., x and y, of cars and people from BDD100K. The gray scale of each square represents the frequency of that coordinate. (a)(b) The spatial distributions of all time. (c)(d) The conditional distributions given that they first appear around the image center in the preceding frame, i.e., temporal distributions.

5.4.2 Spatiotemporal Statistics

Intuitively, spatiotemporal characteristics of different object types are distinctive. From the spatial perspective, a car is more likely to appear at the center of the dashcam image while the pedestrians appear more on either sides. When we consider a dynamic scene where a car is driving at a high speed, a pedestrian on the sidewalk would likely appear closer to the image center as they are far away, and then gradually moves to the left/right side as they get closer. Meanwhile, a car may stay at the image center if it is the preceding car, or it may first appear at the center top but rapidly disappears on the lower left corner if it is an oncoming car.

We verify these empirical observations using a real-world driving dataset, BDD100K [183]. We first extract 2-D histograms of bounding box center coordinates from the BDD100K dataset where videos were recorded from the dash-cam on moving vehicles (See Sec. 5.5.1 for more details of BDD100K). The results are shown in Fig. 5.4. The center coordinates (x, y) of bounding boxes are normalized and recorded as frequencies represented by different gray scales.

Comparing Fig. 5.4a with Fig. 5.4b for single-frame histgorams, we indeed observe that cars incline to be located at the center (of a vehicle camera's field of view) and people are often distributed on either side of the road. Moreover, since BDD100K was collected in the U.S., pedestrians appear on the right side more frequently than on the left, as they are closer to the ego vehicle.

To present the movement patterns of objects over time, we select those objects whose

initial coordinates are at the center, and we show their coordinates in the next fifth frame after the initial one in Figs. 5.4c and 5.4d, for cars and pedestrians respectively. We can observe that cars tend to stay at the center, meanwhile pedestrians (on the side walks) tend to move rightwards because when they are far away they appear at the center but as the ego vehicle drives forward they get closer thus reappear on the right side, and pedestrians crossing the street appears on either sides of the center.

In summary, statistics suggest that there are visual differences of the bounding box behavior of different object categories, which implies that one may use them to distinguish one object category from another.

To formalize this observation, we present probabilistic representation of such spatiotemporal statistics [196, 197] to serve as the foundation of PercepGuard and to motivate our approach. In order to classify a sequence of bounding boxes into an object category, we can use maximum likelihood estimation for classification:

$$c^* = \arg\max_c \quad \Pr(B \mid c) = \arg\max_c \quad \Pr_c(B),$$

where $\Pr(B \mid c)$ means how likely sequence B occurs given that the object belongs to class c. It can be derived as

$$\Pr_c(B) = \Pr_c\left(\{b_i\}_{i=1}^N\right) = \prod_{i=2}^{N} \Pr_c\left(b_i \mid b_{i-1}\right) \cdot \Pr_c(b_1) \tag{5.1}$$

based on Bayes rule and assuming Markov property in that the distribution of the current bounding box depends on the previous one only. The conditional probability $\Pr_c\left(b_i \mid b_{i-1}\right)$ represents the temporal relationship of consecutive bounding boxes, while $\Pr_c(b)$ is the probability of the spatial attribute, i.e., shape and size. Since the above distributions are high dimensional and directly estimating them involves high computational complexity, we choose to use a recurrent neural network to approximate and learn these probabilities.

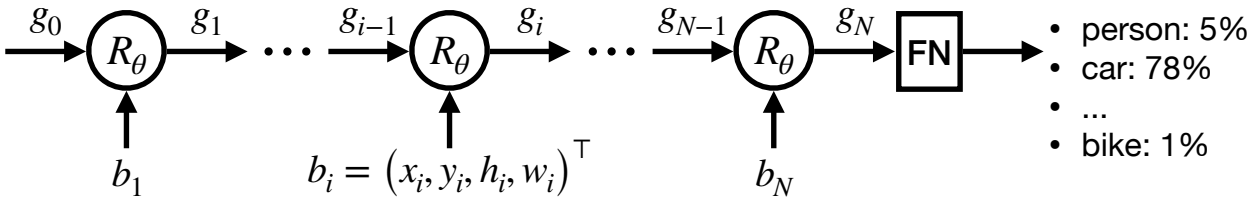


Figure 5.5: RNN-based Sequence Classifier. Bounding boxes are sequentially, and recurrently fed to the RNN layer, R_θ. A fully-connected layer (FN) is then used for classification, with a softmax layer (not shown) for normalization.

5.4.3 Basic Attack Detection Approach

Intuitively, spatiotemporal characteristics of different object types are distinctive. By plotting the emperical histograms from a real-world traffic dataset BDD100K [183] (Sec. 5.4.2), we verify that the bounding boxes indeed show to be statistically informative. We propose to use long short-term memory (LSTM)-based [198] recurrent neural networks (RNNs) to learn the spatiotemporal property of bounding boxes, as they are well-known in sequence classification [199] (e.g., to see whether an English sentence is positive or negative). Note that, our RNN is not intended to be used a standalone object classifier but it rather only serves as a cross validation for an object detector's classification result.

An RNN-based sequence classifier takes a sequence of features as input and outputs the class of the sequence. It is composed of two steps: feature extraction and feature classification. For feature extraction, a typical recurrent operation is actually a feedback loop (Fig. 5.5), which can be described as

$$g_i = R_\theta(f_i, g_{i-1}), \tag{5.2}$$

where f_i is the i-th raw feature vector from the sequence, g_{i-1} is the extracted feature vector calculated from all the previous vectors. Parameter θ is identical for all recurrent operations. Once we have g_N after the entire sequence has been processed, we apply a fully connected layer to g_N for the class confidence vector, $y = C(g_N)$. Further details of the RNN are omitted as they are not required to understand the rest of the chapter.

For our application, we abstract the entire sequence classification process as a function L,

$$y \leftarrow L(B), \quad \text{where} \quad B \in [0,1]^{N \times M}. \tag{5.3}$$

Algorithm 2: PercepGuard Online Attack Detection

Input: b's: Bounding boxes (optionally with contexts)
g_0: Feature initialization vector
θ: RNN parameters
c: Object detector's classification result
τ_F: At least τ_F frames are required
Output: Alarm if attack detected

```
1 t ← 0
2 g ← g_0 ;                                          // Feature initialization
3 while a new bounding box b comes do
4 |   t ← t + 1 ;                                    // Counting frames
5 |   g ← R_θ(b, g) ;                                // Feature extraction
6 |   c' ← arg max_i C(g)[i] ;                       // Feature classification
7 |   if t ≥ τ_F and c' ≠ c then
8 |   |   Raise an alarm
```

Matrix $B = \{b_i\}_{i=1}^{N}$ denotes the sequence of bounding boxes thus $M = 4$ for now and $f_i = b_i$. Here, N is the length of the sequence. The features are normalized into $[0, 1]$ for better classification accuracy. Vector $y = \{y_i : 1 \leq i \leq r\}$ is softmax'ed therefore $\sum_i y_i = 1$, and the classification result is $c' = \arg\max_i y_i$, and r is the number of object classes the RNN model is capable of. The class domains of our RNN model and the object detector are identical in principle.

Our actual detection algorithm runs in an online manner, which is summarized in Alg. 2. Instead of computing (5.3) after obtaining all the bounding boxes for multiple frames, for each new frame, once the object detector outputs a new bounding box for the object we have been tracking, we only need to execute (5.2) once to update the extracted feature g. In this way, we take advantage of the recurrent property of the model to reduce computational time with a minimal space cost of saving the previous g. Finally, we call $C(g)$ to check if its result matches with the object detector's classification result. If not, an alarm will be raised. Our evaluation shows out that as few as five frames (which means 0.17-second assuming 30 fps) are sufficient for classification though we do not put an upper bound to the frame numbers.

5.4.4 Context-based Enhancement

As we will show in Sec. 5.5.3, when we assume an adaptive attacker who also aims to evade PercepGuard, our detection rate drops to 86% (from over 99% against non-adaptive attacks). The reason for such deteriorated detection performance is that, they are able to alter the bounding box behavior which our RNN model classifies into an attacker-intended class. Because the basic model is trained only based on *perceived* object movement which is *relative* to the measurement platform (ego-vehicle)'s movement, the model does not possess high differentiability between cases involving distinct objects that exhibit similar perceived temporal behavior. We show such an example in Fig. 5.8, where two vehicles are following each other at 30 mph (low relative velocity). In order to alter the preceding car into a person, an attacker can merely change its bounding boxes' shapes (from horizontal to vertical) without shifting their locations (keeping the same speed), such that our RNN model recognizes them as a person, because there are plausible cases where the ego-vehicle is moving at a low speed (or stopped) which leads to a low relative speed with a pedestrian.

Therefore, in order to defend against defense-aware attacks, we need more reference/context to validate camera's perception data. There are mainly two sources of reference. First, it can be from other perception sensors such as lidars and radars. Their data can be combined with camera's data via sensor fusion [36, 200, 201]. But similar to a camera, these sensors provide measurement relative to the reference frame (i.e., the sensing platform, the ego-vehicle). Second, it can be the states of the ego-vehicle itself, such as its own speed from VSS', velocity and location from GPS receivers, pitch-roll-yaw positions from IMU, or even environmental factors such as road layout/speed limits and surrounding buildings/terrain from maps, or real-time traffic and weather conditions from live maps (e.g., Apple Maps [202]), etc. They provide side information which leads to more accurate measures of an object's absolute spatiotemporal behavior (as a result of calibrating the relative perception by considering the reference frame), unlike relative measurements output by perception sensors. Interestingly, we will show in our evaluation that relative contexts do not contribute as significantly as those absolute contexts in terms of enhancing adversarial robustness.

To incorporate additional contexts, we extend the feature space of the RNN model to

implicitly learn the relations among all features. In particular, we concatenate all contextual features, denoted as a single vector o, to the original bounding box features, $u = b \,\|\, o$. We use U to denote the sequence of all these features over time, i.e., $U = \{u_i : 1 \leq i \leq N\}$. Then, the sequence classification is

$$y \leftarrow L(U), \quad \text{where} \quad U \in [0,1]^{N \times M}. \tag{5.4}$$

which is similar to (5.3), though M is now larger than four. For example, if we use the ego-velocity $v_e = (x_e, y_e, z_e)^\top$ as context, then $u = (x, y, h, w, x_e, y_e, z_e)^\top$ and $M = 7$.

By incorporating additional context information gathered from other sensors (that provides a more accurate view of the absolute spatiotemporal behavior of an object), the defense-aware attacker is forced to either manipulate the absolute spatiotemporal attributes of the target object (infeasible due to the constraint of overlapping with the perceived location of object), or modify not only the bounding boxes but also the contexts (by compromising additional sensors of different modalities), which is significantly more difficult.

5.5 Evaluation

We evaluate PercepGuard under both adversarial and non-adversarial scenarios with two datasets. We also compare with two baselines, and conduct real-world experiments.

5.5.1 Evaluation Methodology and Setup

We report true negative rates (TNR), true positive rates (TPR), and attack misclassification rates (AMR), defined as the number of misclassified objects divided by the total number of objects, and attack success rates (ASR) which is the number of misclassified objects that is not detected by PercepGuard divided by the total number of objects, i.e., ASR = AMR $\times$ (1 $-$ TPR).

We implement the LSTM-based RNN model as our sequence classifier using Keras [203] with TensorFlow 2.4 [204] as the backend. We use the default architecture of LSTM provided by Keras with 50 memory units. We use Adam optimizer [153] for both training the RNN models and solving the attack optimization problems. We combine with YOLOv3 [1] with

(a) Residential **(b)** Highway

Figure 5.6: Carla Simulated World

SORT [190] as the object detection and tracking system [90] in which we follow SORT's idea for associating objects across frames by matching ground-truth bounding boxes with YOLO-predicted bounding boxes based on IoU.

5.5.1.1 Datasets

Two datasets are used for both scenarios: The BDD100K dataset [183], specifically its multi-object tracking (MOT) subset, contains 1,400 training videos and 200 test videos. All videos are 40-second long and recorded in five frame per second, containing a total of 130K objects with 3.3M of bounding boxes. From the original ten categories, we select five (bike, bus, car, pedestrian, truck) with a total of 5,000 instances due to the imbalance of the original dataset.

For contextual data, we create a dataset collected from the Carla simulator [188, 205], referred as the Carla dataset. In the simulation, a car is driving around in realistic residential areas and highways meanwhile recording video frames of the front scene. There are 1,000 videos, each of which is 200-second long recorded in five frame per second (Fig. 5.6) For each object in each frame, in addition to its bounding box and class, we also recorded contextual data such as the relative velocity and the ego-vehicle velocity. Three categories of objects were labeled: vehicles, pedestrian and traffic signs.

5.5.1.2 Attack Methodology

There are generally two ways to realize misclassification attacks against the perception of an autonomous system. The simplest way is to opportunistically place (e.g., via sticker, projection or display) real images of objects that obscure the victim object [22]. Alternatively, the attacker can adopt optimization-based adversarial ML techniques to generate physically-realizable adversarial patches [94, 206].

The attacker may use an LCD monitor [4] or a projector [22] to add adversarial patches onto the victim object. Either way, they first need to digitally synthesize those patches that can achieve their attack goals, while making sure that the patches can be implemented in the physical world, therefore some physical constraints need to be considered. For example, if they choose to mount an LCD monitor on the back of a car, the physical constraints are the location of the patches (e.g., they cannot be far away from the car), the size (limited by the screen size and attack distance), and the magnitude (limited by the monitor's maximum brightness), etc. We show that despite these challenges, they are still relatively easy to launch, thus bringing severe threats to autonomous systems.

Depending on the attacker's goal, knowledge and capability, there are two types of attackers: defense-unaware and defense-aware attackers (Sec. 5.3.2). Namely, the latter is aware of our detection and tries to evade it while achieving their misclassification goal [207]. From a high level, the adversary aims to find a perturbation sequence Δ to the videos that

$$\underset{\Delta}{\text{minimize}} \quad \|\Delta\| \quad \text{such that} \quad \bar{c} = c'', \quad \bar{c} = c'. \tag{5.5}$$

The attacker minimizes the ℓ_2 norm of Δ to reduce attack cost. The first constraint means that the classification result of the object c'' (by the object detector) needs to be the same as the targeted class $\bar{c}$, i.e., a successful misclassification attack. The second constraint means that the attacker also aims to fool our attack detector to classify the object as $\bar{c}$ as well. Our actual implementation of (5.5) it considers the detail of YOLOv3 and transfers the constraint into the objective function [19]:

$$\min_{\Delta} \|\Delta\| + \mu_{\text{MC}} \cdot \mathcal{L}_{\text{MC}}(X'', \bar{c}) + \mu_{\text{ST}} \cdot \mathcal{L}_{\text{ST}}(X'', \bar{c}), \tag{5.6}$$

where the misclassification (MC) loss measuring how unsuccessful the perturbation Δ is thus far [19], i.e.,

$$\mathcal{L}_{\mathrm{MC}}(X'', \bar{c}) = \max_{x'' \in X''} \left[\max_{\hat{c} \neq \bar{c}} P_j(x'', \hat{c}) - P_j(x'', \bar{c}) \right]. \tag{5.7}$$

$P_j(x, c)$ is the confidence of the class c for the frame x reported by the j-th cell which is the detection cell responsible for the victim object (similar to [5]). It already considers $P_j(x)$, the confidence that the j-th cell contains an object, but we omit it for brevity. The stealthiness loss $\mathcal{L}_{\mathrm{ST}}$ is defined as

$$\mathcal{L}_{\mathrm{ST}}(X'', \bar{c}) = \max_{\hat{c} \neq \bar{c}} L_{\hat{c}}(B'') - L_{\bar{c}}(B''). \tag{5.8}$$

Compare to (5.3) where $L(B)$ returns a confidence vector of all classes, here $L_c(B)$ returns the confidence of Class c for a given bbox sequence B. Of course, B can be replaced with a feature sequence U when context is considered (Eq. 5.4). Such formulation of adaptive attacks is in fact commonly used, e.g., in [96]. In the formuation of defense-unaware attacks, $\mu_{\mathrm{ST}} = 0$.

We follow the method proposed by Brown et al. [94] to emulate the patch. We initialize the patch in the same size of the benign frames and then transform it into a smaller region where the object of interest resides to emulate the physical attacks. We vary the patch size in our simulation. To overlay patches onto to benign frames, we follow Carlini and Wagner's tanh method [19] to guarantee a valid range of pixel values. Note that, simulated attacks are stronger than physical domain attacks since the former assumes an ideal sensing channel.

5.5.2 Non-adversarial Scenarios: True Negatives

In non-adversarial scenarios where there is no attack, we aim to test PercepGuard's performance in terms of true negative rates because too many false alarms may cause the autonomous system to malfunction.

BDD100K Dataset Results In MOT, the lengths of object bounding box sequences are different because they appear in the field of view for different durations. We have two choices when training the RNN model: We could either truncate them into a fixed length, or leave them in variable lengths. In the case of fixed-length, k-gram sequences, for all pairs of

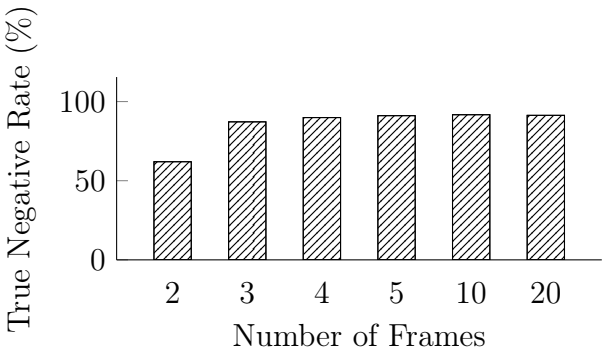
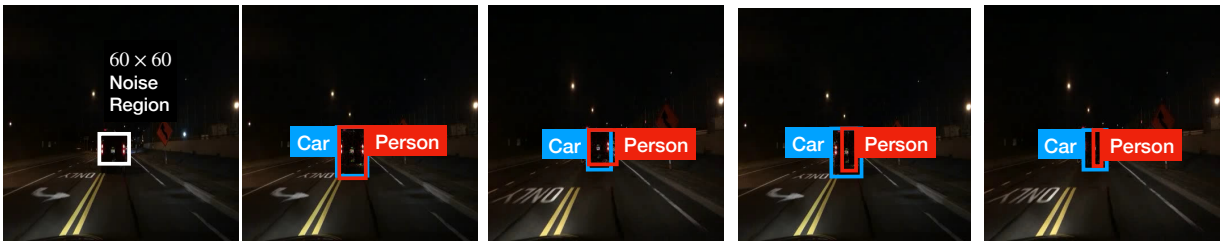


Figure 5.7: True negative rates v.s. number of frames

data points, $\{(B, c)_i\}$, the size of B is (k, M), where $M = 4$ since the dataset only provides bounding boxes without other context. In the variable length case, we simply feed the raw, variable-length B's to the RNN model. The disadvantage of this method is that it takes longer to train as it doesn't vectorize well. However, at test time both methods perform equally fast. Results show that the LSTM model trained with variable-length traces performs better in terms of classification 5-fold cross-validation accuracy: 97% v.s. 92% (4-gram). We also confirm that similar differences can be derived from the IMDB dataset [208]. Thus, we adopt the variable-length training approach for the rest of the chapter.

To test the model, we use YOLO's bounding box predictions instead of the manual labels that we used to train the model as there won't be manual labels in actual deployments. For reference, YOLOv3's mean Average Precision is 55.3. Results show that the test accuracy is 95%, which means the false positive rate is only 5%. The test accuracy is slightly lower than the validation accuracy (97%) because YOLO's bounding box predictions are unstable. We also evaluate how many frames does PercepGuard require to make accurate classification. Fig. 5.7 shows that the true negative rate saturates at around four or five frames, which means PercepGuard needs as few as five frames (with 30 FPS, it is 0.17-second) to make a classification decision.

Carla Dataset Results We adopt the variable-length training method. The 5-fold cross-validation accuracy is 95%. When testing the model, we run YOLO on Carla's simulated images for the bounding box predictions. The test accuracy of our RNN model on these predictions is 92%, indicating a false positive rate of 8% which is higher than the BDD100K;



(a) Noise Region **(b)** $t = 1$ **(c)** $t = 2$ **(d)** $t = 1$ **(e)** $t = 2$

Figure 5.8: A video from BDD100K where the defense-unaware attack fails to evade PercepGuard's detection but the defense-aware attack succeeds. These bounding boxes are recreated based on actual predictions for better presentation. (a) The vehicle at the image center is overlaid with an adversarial patch. (b)(c) The defense-unaware attackers effectively alters the classification result of a car into a person but the bounding boxes are left similar to a car's bounding boxes, therefore the attack is detected by PercepGuard as the tracker and the class predictions are inconsistent. (d)(e) On the other hand, since the defense-aware attackers considers PercepGuard when they are solving for the optimal adversarial patches, the optimizer finds those patches that not only alters the classification results, but also "shrinks" the bounding boxes to be vertical (as if a person's bounding box would be) so that PercepGuard regards the tracker to be consistent with the object class.

this is because the official, pre-trained YOLO model was trained on the COCO dataset [162], a real-world image set whose pixel distribution is different than the Carla-simulated pixels.

Since our Carla dataset also provides rich contextual data, we select two contexts, relative velocity and ego velocity, along with bounding boxes to train and test our RNN model to see whether they can help improve the true negative rates. We added random noise drawn from the standard Gaussian distribution, $\mathcal{N}(0, 1)$, to those contextual data in order to simulate natural sensor errors (compared with 0-60 MPH, the range of those velocities). Results show that the improvement is minor, with a test accuracy of also roughly 92% regardless of which velocity/velocities we append to the bounding box features. This is because those additional features do not provide extra information for sequence classification; however, we will show that they become useful in adversarial scenarios where the consistency among features is more critical.

5.5.3 Adversarial Scenarios: True Positives

Two types of attacks (Sec. 5.5.1.2) are evaluated: defense-unaware attacks and defense-aware attacks (a.k.a. adaptive attacks). For demonstration purposes, we assume the attacker aims to alter a "car" into a "person" as a representative, severe case which could result in a

hard brake executed by the victim vehicle on a highway.

We use the Adam optimizer [153] to solve attack optimization problems with $\mu_{\text{MC}} = 100$ and $\mu_{\text{ST}} = 100$, which means we would like the optimizer to focus more on the misclassification and stealthiness than minimizing the perturbation magnitude[2]. The optimization process terminates when it converges or reaches the maximum number of iterations, which is set to 1,000 steps. The learning rate is 0.01. After its termination, we check both losses: If $\mathcal{L}_{\text{MC}} \leq \tau_{\text{MC}}$, we mark the object as misclassified; If $\mathcal{L}_{\text{ST}} \leq \tau_{\text{ST}}$, we mark this attack as stealthy as it has evaded PercepGuard's detection. We let $\tau_{\text{MC}} = 0$ and $\tau_{\text{ST}} = 0$ for the digital domain (simulated) evaluation as it is easier for the attacker to achieve. However, we raise them for real-world evaluation (Sec. 5.5.5) for higher attack confidence due to the random environmental factors that make the attack not as stable as in simulation.

BDD100K Results We first present the results on the BDD100K dataset (Table 5.1). We vary the size of adversarial patches from 20×20 to 60×60 pixels to emulate various attack distances and patch display sizes, as compared to the input image size for YOLOv3 which is 416×416 pixels. As the patch size increases, the attack performance generally gets better because there are more pixels upon which the attacker can manipulate. For defense-unaware attacks (Table 5.1) specifically, the AMR starts from 83% when the patch size is only 20×20, and approaches 93% at 60×60, which means the attack is considerably effective, indicating a severe threat to autonomous vehicles. However, when we feed to PercepGuard the bounding boxes of those compromised objects that are misclassified as "person" by YOLOv3, nearly all of them are (correctly) classified into their benign class "car", i.e., the TPR is almost 100%. As a result, almost all the attacks are unsuccessful, i.e., the ASR is nearly zero.

On the other hand, defense-aware attacks (Table 5.1) yield lower AMRs, because in this case the attacker aims for two objectives: misclassification and stealthiness for which the noise can contradict with each other. In other words, part of the gradient descent is made towards stealthiness, compared with the defense-unaware attacks where almost all of the gradient descent is for misclassification. The TPRs of PercepGuard are slightly lower because

[2]Ideally, both constants should be adjusted via binary search in order for them to be as small as possible [19]; however, that is only for norm-bounded perturbation which is not assumed in this chapter, as it is too weak to test against PercepGuard.

Table 5.1: Adversarial patch attacks with BDD100K

Attack Type	Patch Size	AMR	TPR	ASR
Defense-unaware	20×20	83.47%	99.63%	0.3%
	40×40	89.41%	100%	0%
	60×60	92.94%	100%	0%
Defense-aware	20×20	73.25%	98.74%	0.92%
	40×40	80.49%	90.33%	7.78%
	60×60	87.6%	85.67%	12.55%

such an attacker aims to evade the detection by optimizing the adversarial noise to modify the object class and bounding box behavior (e.g., shape and locations) at the same time. Without any additional context, although the RNN classifier can distinguish different object classes when the attacker does not intentionally modify the bounding box behavior, there can be boundary cases where two objects of different classes appear to have similar behavior (which is also why there are false positives). For example, a vehicle that is turning at an intersection may see a person on a sidewalk (with bounding box shape being modified by an attacker) as similar to a car crossing the street, which can be confused with the case of seeing a real vehicle crossing the intersection when the ego vehicle is waiting for a red light. The defense-aware attack may exploit such corner cases in the confusion matrix, which may only require changes in bounding box shapes but not their locations. Note that the latter is more difficult to achieve as the attacker only apply patches within the region of the victim object. Even with a slightly lower TPR, we would like to point out that for a 416×416 image, a 60×60 patch is large enough to entirely occlude a car from the back for most instances in BDD100K.

In summary, the defense-unaware attacks are highly effective, but PercepGuard is able to detect most of them with a detection rate as high as 100%. Adaptive attackers are able to bypass our basic detection approach for at most 12.55% of their attacks but note that such attacks are fairly impractical to realize. Nevertheless, next we show that contextual data are able to help us strengthen the robustness of PercepGuard.

Carla with Context We use our Carla dataset with contextual information to evaluate the same adversaries to show how context can help enhance PercepGuard. We only consider

Table 5.2: Differing attack capabilities at 60×60 patch size against RNN trained with contexts

Compromised Sensors	AMR	TPR	ASR
Camera Only	88.76%	99.35%	0.6%
Camera + LiDAR	89.90%	97.88%	1.9%
Camera + VSS	90.47%	85.26%	13.33%
Camera + VSS + LiDAR	90.85%	72.74%	24.76%

defense-aware attacks here as they are stronger.

For demonstration, we select two contexts, ego velocity and relative velocity, which can be obtained from vehicle speed sensors (VSS) and LiDAR/radar, respectively. We train the RNN model following the form of (5.4), where $M = 10$ because each velocity is represented in 3-D. Here, we assume a strong attack model where the attacker can compromise different sensors simultaneously, and more importantly they can synchronize each sensor attack precisely.

Table 5.2 summaries the attack and defense performance with different attack capabilities, namely which sensors are compromised. When comparing the first row of Table 5.2 with the last row of Table 5.1 where only the camera is compromised, the misclassification rate is similar but TPR is increased from 86% to 99% which makes the attack success rate approach zero; this is because the RNN model has learned the consistency among those ten features during training thus able to detect the inconsistency of the tampered bounding boxes, therefore still classifies the feature sequence as a car for 99.4% of the cases.

When we compare the second and the third rows of Table 5.2, where the attacker can additionally compromise LiDAR or VSS sensors, we observe that our detection rate decreases more when the VSS is compromised: 85% vs 98% (for compromising LiDAR), although both velocities have an identical number of features (three scalars). This suggests that the ego velocity plays a more important role in PercepGuard's consistency verification, because relative velocities can be implicitly inferred from bounding box size changes (e.g., when the front car gets closer, its bounding box gets larger). In fact, such size changes have been used to estimate the following distance [209]. In other words, existing LiDAR attacks [80, 82] would not have a large impact on PercepGuard's performance because LiDAR's outputs are not as critical as VSS', and yet the VSS attack is more difficult to conduct as it requires close proximity (a few centimeters) to the victim vehicle [141].

Finally, if the attacker can compromise three sensors at the same time, PercepGuard can still detect 73% of the attacks. The reason why it yields a lower detection rate is that when the capacity of a machine learning model (e.g., a neural network in our case) increases, it becomes more susceptible to adversarial attacks [20]. At a glance, this is worse than not using contextual data (Table 5.1). However, we argue that such an attack is extremely difficult to launch under our threat model as it requires the attacker to gain the white-box knowledge of three different systems and meanwhile synchronize three separate attacks precisely. Another way to achieve such a strong attack would be to remotely hack into the CAN bus of a vehicle, where an attacker digitally alters the sensor data before or after the processing algorithms [44]. However, such an attacker is not considered in our threat model.

Model Sensitivity We study PercepGuard's sensitivity to evaluate its vulnerability to attacks that are outside of our threat model, namely ones that can directly modify the input to the RNN, and ones that can perturb larger areas on the images. We show that PercepGuard is sensitive to neither of them.

We evaluate PercepGuard's sensitivity to subtle perturbation to its input, i.e., the bounding boxes to see if it would be vulnerable to more advanced perception attacks that might be developped in the future. As we will see from the results, PercepGuard is not sensitive to small perturbation of bounding box features (Sec. 5.5.3.1), nor to attacks with larger perturbation areas on the images (Sec. 5.5.3.2).

5.5.3.1 Bounding Box Perturbation

We evaluate the sensitivity of the RNN used in PercepGuard. We directly perturb the input to the RNN, i.e., bounding box sequences, to see how much perturbation is needed to change the classification result. The optimization formulation is

$$B^* = \arg\min_{B'} \quad \mathcal{L}_{\mathrm{ST}}(B', \bar{c}) + c_{\mathrm{IoU}} \cdot \mathcal{L}_{\mathrm{IoU}}(B, B'), \tag{5.9}$$

where B is the benign bounding box sequence, $\bar{c}$ is the target class, and $\mathcal{L}_{\mathrm{IoU}}$ is the intersection over union loss that measures how close B is to B'. The stealthiness loss $\mathcal{L}_{\mathrm{ST}}$ indicates how

likely the RNN classifies B' as $\bar{c}$, defined as

$$\mathcal{L}_{\mathrm{ST}}(B, \bar{c}) = \max_{\hat{c} \neq \bar{c}} L_{\hat{c}}(B) - L_{\bar{c}}(B).$$

Recall that $L_c(B)$ is the confidence that the RNN classifies B as c. To find the minimal perturbation to B, we follow the strategy from [19]: We start $c_{\mathrm{IoU}} = 1$ and see if $\mathcal{L}_{\mathrm{ST}}(B', \bar{c}) < 0$ when the optimization process converges. If not, we increase c_{IoU} and try again, until $\mathcal{L}_{\mathrm{ST}}(B', \bar{c})$ falls below zero.

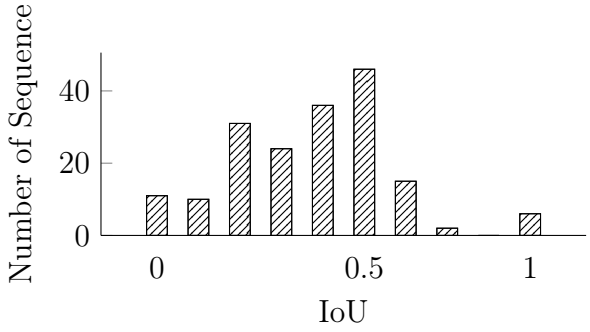
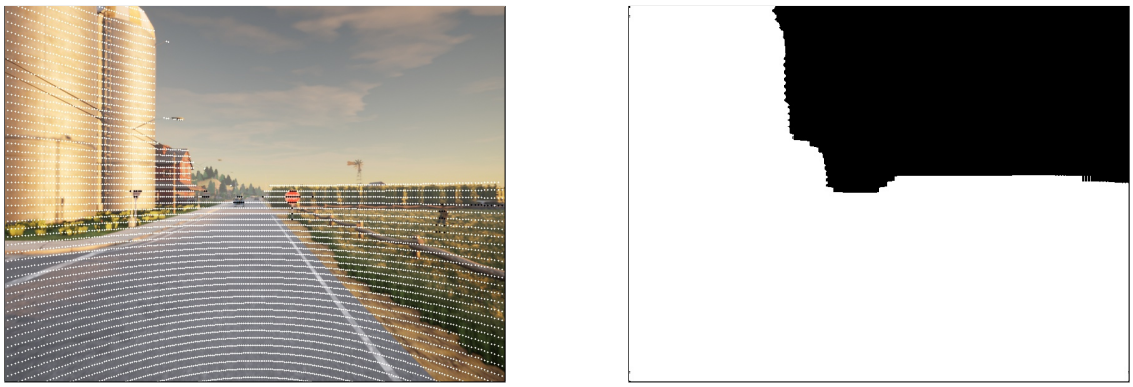


Figure 5.9: IoU Histogram for direct perturbation to bounding box sequences

We compute (5.9) for 181 sequences of bounding box of length ten, and the IoU histogram is shown in Figure 5.9. We can see that over 87% of the bounding box sequence need to be resized and/or shifted significantly, with the IoU less or equal to 50%, which means in order to change the classification result of the RNN, the attacker needs to alter the bounding boxes significantly (with IoU more than 50%).

5.5.3.2 Larger Perturbation Area

Previously, we evaluated adversarial patch attacks with sizes up to 60×60 which is limited by the size of the monitor. Here, we relax such a limitation by simulating that the attacker uses a projector to project adversarial noise on to the front scene of the victim vehicle, similar to the Phantom attack [22]. We assume that the projector is powerful enough to perturb any objects other than the sky. In this case, the perturbation area occupies over 50% of the pixels. We use image segmentation to identify the perturbable areas for the BDD100K dataset and use LiDAR points for our Carla dataset (Figure 5.10).



(a) RGB Pixels and Lidar Points **(b)** Perturbation Mask

Figure 5.10: From the Carla dataset, the perturbation mask is based on the LiDAR points. The white part of the mask indicates the perturbable area while the sky cannot be perturbed. The masks for BDD100K are similar but based on image segmentation.

Table 5.3: Attacks with larger perturbation areas.

Dataset	A.M.R.	TPR	ASR
BDD100K	88.08%	42.35%	62.7%
Carla	90.67%	96.91%	1.9%

Results from two datasets are shown in Table 5.3. When comparing with the last row of Table 5.1, the attack misclassification rate (A.M.R.) for BDD100K is just slightly higher even though the perturbation area is now much larger. This is because perturbing the pixels that are not on the vehicle has little impact in altering its classification result. However, the true positive rate (TPR) drops since now the attacker has more room to shift the bounding boxes in order to bypass our spatio-temporal consistency check. This result also shows that our attack is powerful. When considering the contexts provided by the Carla dataset, our TPR increases to 96.91% because the additional contexts that the attack cannot perturb, provide more evidence to detect the spatio-temporal inconsistency of the shifted bounding boxes.

Model Transferability Since manual labeling is expensive but simulated dataset can be scaled easier, we evaluate the transferability of our RNN model here. In particular, we train the RNN model using the Carla dataset and test it on BDD100K. Results show that the true negative rate is 83%, and the true positive rates against defense-unaware attacks and defense-aware attacks are 92%, 87% respectively for 20×20 patches in simulation. Although

these rates are not significant compared with Table 5.1, these results still suggest that one could train the RNN model on large-scale, simulated dataset and then fine-tune it with small-scale, real-world dataset [210].

5.5.4 Baseline Comparison

We experimentally compare PercepGuard with NIC and SCEME as baselines (Sec. 5.5.4), since they are both also input-agnostic, and SCEME also uses contextual consistency.

NIC [46] detects adversarial examples against image classifiers by checking the network invariant. It produces a binary decision for each individual video frame. SCEME [10] also works on individual frames but it produces a scalar for each object by verifying its contextual consistency with other objects, backgrounds and scenes within the frame. See A common drawback of both is their limited applicability: NIC is designed for image classification only, which is not as commonly used in autonomous driving as object detection. Though its principle may be applied to object detection models but it needs to be significantly redesigned. SCEME relies on RPNs thus being fundamentally incompatible with one-stage algorithms such as YOLO that is more commonly used in autonomous driving [15, 90].

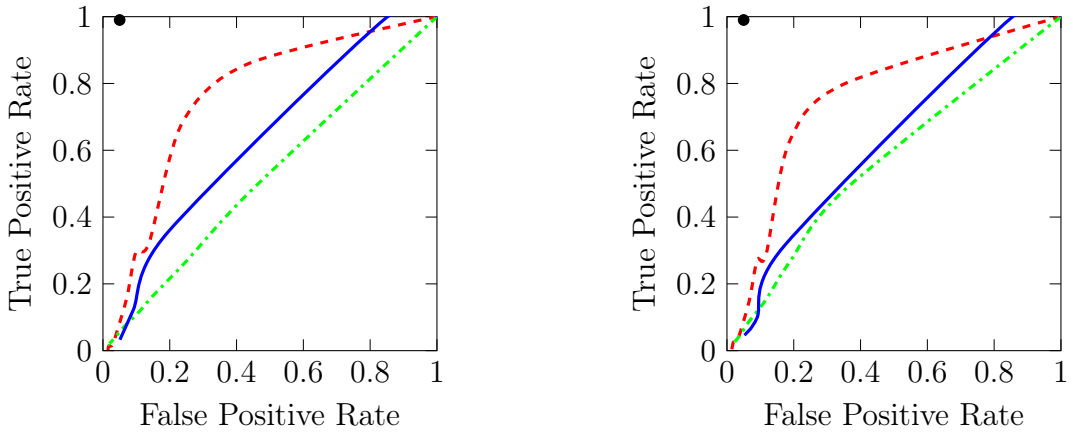
Most importantly, both methods do not consider temporal features across multiple frames. Let us elaborate conceptually why temporal features are useful in object misclassification detection. Consider this scenario where an object labeled "person" is located at the image center, which means it's in front of the ego-vehicle. SCEME is unable to distinguish whether the object is a real person who is walking across the street with the ego-vehicle waiting for them, or a preceding vehicle that is misclassified as a person, since it only focuses on individual frames. On the other hand, PercepGuard can identify the difference since it additionally considers temporal features and contexts: If the object is moving from side to side slowly while the ego speed is nearly zero, it's a person; If it stays at the image center while the ego vehicle is driving, it's a misclassified vehicle.

Methodology Since both of them output detection scores for a single frame, we need to first integrate inter-frame scores into a single score for a sequenc of frames. Given a sequence of detection decisions, $\{s_i; 1 \leq i \leq N\}$ with s_i being one (adversarial) or zero (benign),

there are two integration criteria: (1) By *portion*: The percentage of frames that are marked adversarial, i.e., $\sum_i s_i/n$, and (2) By the length of the longest *consecutive* frames that are marked adversarial. For both criteria, if the integrated score s exceeds some threshold we mark the whole sequence as adversarial. We present receiver operating characteristic (ROC) curves with varying thresholds for both NIC and SCEME, to compare with PercepGuard's performance which is a single point in the ROC plots (because in PercepGuard, the RNN's classification result is the class with the highest confidence, therefore there is no threshold for PercepGuard).

Considering the differences of design and applicability between NIC and PercepGuard, to compare NIC with PercepGuard we crop out the portion of the object of interest from the original frame in the BDD100K videos and use the crop-out images to form a benign dataset that has ten object classes. We use the dataset to train two image classifiers in two architectures, DenseNet [211] and the one from [19] (same as [46]). Both classifiers achieve test accuracy higher than 98%. Then, we train NIC on each of them. At test time, for each perturbed video (defense-unaware attacks with either adversarial patch or real-image) from Sec. 5.5.1 we obtain a sequence of crop-out images $\{x'_i : 1 \leq i \leq N\}$ and feed each x'_i to the classifier and its corresponding NIC for which we obtain a sequence of binary decision $S' = \{s'_i : 1 \leq i \leq N\}$ for TPR. Similarly, for each test benign video we obtain a benign score sequence $S = \{s_i : 1 \leq i \leq N\}$ to compute FPR.

Due to SCEME's incompatibility with YOLOv3, in order to compare it with PercepGuard, we conduct new experiments to evaluate PercepGuard with Faster-RCNN. We use the MOT dataset for the evaluation. From each video, for each "car" or "person" object across the N frames we obtain a sequence of its context profiles $U = \{u_i : 1 \leq i \leq N\}$, as well as a sequence of its bounding boxes $B = \{b_i : 1 \leq i \leq N\}$. Then, we use the IFGSM algorithm [10, 212] to perturb the "car" objects in each frame so that classifies them as a "person". Similarly, we collect the context profile sequence U' and the bounding box sequence B' of each object that is successfully misclassified. To train the autoencoders, we gather all context profiles u's (in a total of 10K per class) of benign objects and divide it into a training set (80%) and a test set (20%). We use the same hyper-parameters as in [10] for the training. We put together the benign test profiles and the adversarial profiles u' to get the threshold of the reconstruction



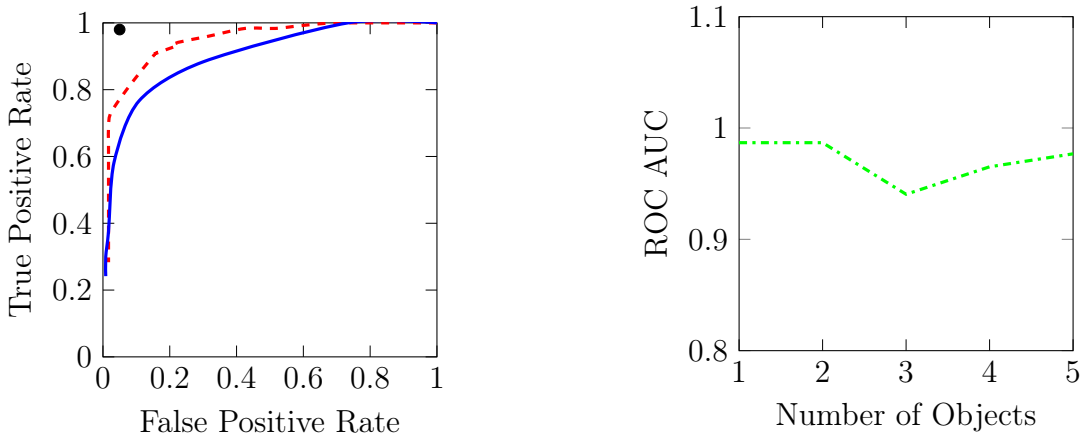
(a) NIC with Carlini Net **(b)** NIC with DenseNet

Figure 5.11: ROC curves for NIC with two different network architectures. Real-image attacks are tested with the portion criterion (-·-·-). Adversarial patch attacks are tested with both the portion criterion (- - -) and the consecutive criterion (——). PercepGuard's performance is marked with •.

loss, τ_{RL}, at the equal error rate. With this threshold, we can convert a sequence of context profile into a sequence of binary detection. To test PercepGuard's performance, we feed the bounding box sequences in the benign case B to our RNN for the FPR, and adversarial sequences B' for the TPR.

Results Following those two inter-frame integration criteria, we calculate the ROC curves (Figure 5.11a) based on benign score sequences and adversarial score sequences. The threshold for the portion criterion varies from 0% to 100% and the threshold for the consecutive criterion is from two to six. Overall, PercepGuard performs better than NIC in all cases mainly due to the fact that NIC is designed for image classification while the video frames used at test time are perturbed against object detection. Furthermore, NIC is unable to detect real-image attacks because the noise are not generated via optimization-based methods, which do not cause the network invariants to be out-of-distribution for NIC to detect. Both inter-frame integration criteria produce a worse trade-off between TPR and FPR than PercepGuard does because PercepGuard takes advantage of the temporal consistency meanwhile NIC subjects to its detection performance on individual frames.

From the ROC curves in Figure 5.12a, we can see that SCEME works significantly better than NIC (Fig. 5.11a). This is because SCEME is designed for object detection but NIC is



(a) SCEME **(b)** Varying Object Numbers

Figure 5.12: (1) ROC curves of SCEME. (2) SCEME's performance (-·-·-) depends on the number of objects in the scene, which is consistent with the original paper of SCEME [10].

designed for image classification. Also, the proportion criterion outperforms the consecutive one which is similar to NIC's results. PercepGuard achieves 98% true positive rate and only 5% false positive rate, which is close to the result from the experiments with YOLOv3. This shows that PercepGuard can be applied to both one or two-stage object detection algorithms and both attack methodologies (IFGSM [212] and C&W [19]). We also show that SCEME's performance subjects to the richness of the contexts (Fig. 5.12b).

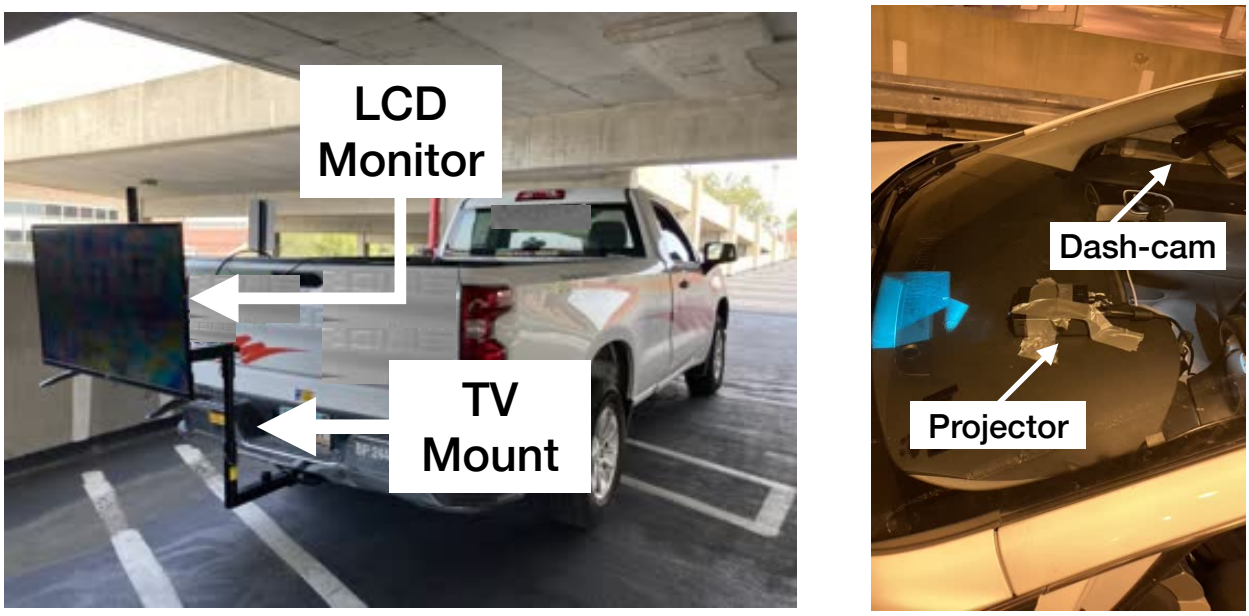
Because SCEME relies on the context on the scene, we anticipate that the richness of the context (e.g., object-object spatial/coexistence relationship) effects its performance, meanwhile PercepGuard's performance does not subject to such variation since it relies on the contexts of which the richness is more stable. To verify this, we divide the profile sequences into groups and in each group the average number of objects over the frames in every sequence is identical. For each group we calculate the area under curve (AUC) of the ROC curve, and we plot the AUCs in Fig. 5.12b. We can see that SCEME's performance first drops and then arises as the number of objects increases, which is consistent with the result in Fig. 6 from the SCEME paper due to the balance between object-object context and the rest of contexts [10]. However, PercepGuard performs more stably with varying numbers of objects, with true positive rates above 98% and false positive rates around 5%.

Model Extensibility Our framework is general such that it can be easily extended to work with additional contextual features as long as they contain spatiotemporal attributes (e.g., 3-D bounding boxes, point cloud [80], etc.). We combine SCEME with our RNN to see if there is spatio-temporal consistency from the context profiles outputted by SCEME. Recall that for each object in each frame, SCEME outputs a context profile $u = [r, \gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$. We run SCEME on the MOT subset of the BDD100K dataset, and for each object we obtain a sequence of its context profile $U = \{r_i : 1 \leq i \leq N\}$ where $N = 5$. We select four object classes: bike, bus, car, and pedestrian (missing truck because the context Faster-RCNN used by SCEME is trained with the VOC dataset [213] that do not include trucks.) We collect a total of 10K of such sequences and use 80% of them to train the LSTM with 1000 memory units (the size of a context profile u is 5×4096), and 20% for testing. The 5-fold cross validation shows the test accuracy is 97% (false positive rate is 3%). To test the attack detection performance, we feed U', the sequences of context profiles of attack objects, to the RNN classifier and the true positive rate is 98% based on 200 sequences. This means that the framework of PercepGuard is compatible with features that are not human-interpretable. We underline that we only use the r part of the entire context profile u; we test the cases where we either use only one of the remaining feature vectors (γ, the GRU gates), or concatenate all of them together with r: the test accuracies are all below 40%. The reason may be that r is the final fused result which contains much richer contextual information than the gates.

5.5.5 Real-world Experiments

We conducted a series of real-world experiments to demonstrate the threat of adversarial patch attacks, and more importantly, to evaluate PercepGuard's performance in real-world environments and under realistic attacks.

Methodology and Setup To implement the adversarial patches, we extend the experiments done by Hoory et al. [4], where we mount an LCD monitor [11] on the back of a vehicle (Fig. 5.13) using a tailgate TV mount [214]. Hence, we are able to drive around with the LCD while Hoory et al. only performed stationary tests, which means our setup is more practical than theirs. We additionally use a portable projector [13] mounted on the dashboard



(a) Front vehicle **(b)** Rear vehicle

Figure 5.13: Real-world experiment setup. (a) An LCD monitor [11] is mounted on the back of the front vehicle. (b) A dash-cam [12] is mounted on the windshield of the ego vehicle. A portable projector [13] is taped on the dashboard.

of the ego vehicle to replace the LCD monitor as a different noise injection vector. The experiments were conducted in a big, campus parking garage during both daytime and night, and a campus street only at night because during daytime both the monitor and the projector would be completely overwhelmed by the sunlight. For the monitor experiments, we reduced its brightness at night, because otherwise the content would be saturated. We collected 45 videos in total.

We regard safety as the highest priority. We carefully chose the location and time to conduct the experiments with minimal traffics. We chose holidays and weekends when there were not many pedestrians and vehicles in the garage or on the street. The experiment was setup and tested on the highest level of a parking garage where there was no driving vehicle or pedestrians. Once we confirmed the setup was secured (e.g., by wiggling the vehicle a little bit), we moved on to the lower levels and then to the streets for which the route was carefully planned such that the streets were not at all crowded (to be reworded), the speed limit is 20 MPH or below (to be confirmed), and there was only one lane for each traffic direction. We didn't stay in one street for too long to avoid blocking normal traffics since our speed was 5

MPH below the speed limit.

To make sure the LCD monitor was mounted on the vehicle securely, we used a tailgate TV hitch mount and used a pickup truck that had a hitch for the mount to attach. We made sure each joint of the mount was fastened tightly before each round of experiments. We additionally used bungee cords and duck tapes to prevent any relative movement of the monitor to the truck (such as panning or tilting). They were also used to organize and secure the cables.

Both vehicles were fully operated by experienced human drivers with no autonomous driving module installed (since the experiments were conducted only to collect data for offline test). We avoided aggressive acceleration, deceleration and turning on the road to further minimize the chance of the monitor falling down. The driver in the following vehicle paid extra attention to the monitor for any sign of instability. At any time, the following vehicle was directly behind the preceding vehicle without other vehicle intercepting, and the phone communication between two drivers was maintained, so that in case of any emergency, such as the monitor fell off for which both vehicles could stop immediately without any vehicle rolling it over. The experiment executors are experienced with similar outdoor experiments with vehicles in motion. The experiments were supervised by professors who have many years of experience in autonomous vehicle research. Traffic rules were obeyed strictly.

The victim vehicle is driving behind with a dash cam [12] installed on its windshield acting as the vision camera of an autonomous vehicle [16], that records videos of the front scene. The dash-cam is equipped with a GPS receiver so it stamps vehicle speeds on the frames. We manually take those speeds out and transfer them into 3-D velocities with two other directions being zeros. We use the RNN model trained with our Carla dataset as PercepGuard's detection classifier in order to utilize the context of speed for consistency verification. If the attacker makes the object misclassified for three consecutive frames, we count it as an effective attack.

Real Images We first implement the opportunistic attacks (without any optimization), by using both devices to display some real images of people and stop signs on the back of the front vehicle (Fig. 5.14) to see if YOLO can recognize them, and more importantly, to

(a) Person **(b)** Displayed STOP **(c)** Projected STOP **(d)** Adversarial patch

Figure 5.14: Real image attack examples. Bounding boxes recreated for better presentation. (a)(b) Both the person/stop-sign images and the truck are recognized by YOLOv3. (c) Projected stop sign obscured by the truck's paint thus not detected by YOLOv3. (d) The adversarial patch alters the truck into a person.

Table 5.4: Real image attacks in the real-world

Real Images of	Device	ARR	TPR	ASR
People	Monitor	63.2%	83.3%	10.6%
	Projector	58.8%	100%	0%
Stop Signs	Monitor	40.0%	100%	0%
	Projector	20.0%	100%	0%

see if PercepGuard can detect them with the speed context. In general, the monitor yields higher attack recognition rates (ARR, for which YOLO classifies them as intended by the attacker) than the projector does (Table 5.4); this is because when the following distance changes, it is challenging to change the projector's focal length in real-time as it requires manual adjustment by the driver (who in our experiments also needs to operate the vehicle simultaneously). Another reason is that the projection screen is unstable while the monitor is fixed on the truck. More interestingly, the ARR of stop signs are lower than that of a person. Because we only displayed the top of a stop sign without the pole, one explanation is that YOLOv3 also takes objects' surroundings into consideration [63, 152] which in this case is the pole. PercepGuard is able to detect most of the attack instances (except two pedestrian cases). We manually checked those failed cases; the reason is that both vehicles were driving at only 2 MPH, which PercepGuard accepts because it is also a reasonable speed for pedestrians.

Table 5.5: Adversarial patch attacks in the real-world

Attack Type	Device	AMR	TPR	ASR
Defense-unaware	Monitor	52.2%	100%	0%
	Projector	27.3%	100%	0%
Defense-aware	Monitor	45.5%	100%	0%
	Projector	20.0%	100%	0%

Adversarial Patches Next, we implement the (optimized) adversarial patch attacks in the physical/perception domain and conduct experiments with the following procedure. We first drove around the area with one car following another at various distances from one to ten meters during daytime and at night, meanwhile capturing benign frames without the LCD. Then, we fed those benign frames into the optimization processes (Sec. 5.5.1.2) to solve for 20×20 (because it is a 40-inch monitor) adversarial patches that alter a "truck" into a "person". We specify the pixel sizes and coordinates of the patches so that they can be realized by both attack devices. Because the color accuracy of the dashcam and both devices is imperfect, to compensate for channel effects, we manually calibrated the color by making the red and green channels twice as stronger and the blue channel weaker by half. Next, we installed the monitor/projector and displayed those optimal adversarial patches (with calibrated color), at the same time we drove around the same area while recording adversarial frames. Finally, we analyze these adversarial frames for attack misclassification rates, and PercepGuard's true positive rates.

Results in Table 5.5 show that defense-unaware attacks yield a higher misclassification rate but PercepGuard is able to detect all the attack instances. This is partly because the physical noise makes the bounding box predictions more unstable, or bouncier, than the instances from simulations (Secs. 5.5.2 and 5.5.3). Especially, in the defense-aware attacks, the attacker tries to control the bounding boxes to fool PercepGuard, but to the contrary, they became even more unstable in the real-world.

Attack Cost and Difficulty To achieve higher ASR, the attacker needs to increase the attack cost for acquiring larger and brighter LCD monitor/projector and powering them. E.g., to realize a 60×60 patch a 120-inch monitor is required which is projected to be $40K.

Moreover, additional contexts further raise the cost and difficulty, as they need not only the knowledge of those different sensor systems, but also the equipment to spoof those sensors; let alone synchronizing each individual attack precisely.

First, for the monitor-display based attack method, we conducted the LCD monitor experiments with a 20×20 patch size because it is a 40-inch monitor at a distance of five meters. In order to achieve 60×60 as we simulated where the attacker has non-zero success rate (Table 5.1), they need at least a 120-inch display for outdoor. For reference, 50, 55, 75-inch outdoor displays are roughly \$3.2K, \$3.8K and \$10K [215]. Based on such an exponential trend, a 120-inch monitor could be \$40K.

Similarly, projectors with higher brightness are more expensive. Empirically, the luminance required for a projector is proportional to the noise magnitude ℓ_∞, ambient illuminance, distance square, i.e., $\text{Lumen} \propto \ell_\infty(\Delta) \cdot \text{Illum} \cdot d^2$. From our real-world experiments at night with an illuminance of 30 lx, our \$180, 300-lumen projector [13] was able to induce $\ell_\infty = 0.1$ at a distance of two meters. In order to bypass PercepGuard where typically an ℓ_∞ norm of one is required, the attacker needs a 9K-lumen one priced at \$13K [216] at night at a distance of three meters. During daytime when the illuminance is 40 klx, one may need a 75K-lumen projector for which a used one is priced at \$375K [158].

Not to mention these attacks can easily be thwarted using additional contexts (Table 5.2). In terms of evading our context-enhanced defense, to attack multiple, heterogeneous sensors simultaneously, the attacker needs not only the knowledge of those different sensor systems, but also the equipment to spoof those sensors [36]; let alone synchronizing each individual attack precisely. In summary, PercepGuard significantly increases the cost and difficulty for an attacker to evade the detection.

5.6 Discussion

We discuss the potential applicability, extensibility, and limitations of PercepGuard.

5.6.1 Other Types of Attacks

Although PercepGuard focuses on detecting misclassification attacks, it can potentially detect object creation attacks [6, 9, 22] and tracker hijacking attacks [90] because the attack may cause the spatiotemporal inconsistency. The former creates a non-existent object by exploiting a variety of modalities (camera, LiDAR, radar, etc). For example, attacks against camera sensors include injecting noise directly into the camera sensor (e.g., using visible light projection [6, 7], radio-frequency electromagnetic signals [187], etc), and exploitation of imperfections of camera sensors (e.g.,the ghost/flare effect [6, 7] and rolling shutter effect [8]), or projecting an image on environmental surfaces (e.g., sidewalks or walls [22]). However, our combination of detection heuristics and imposition of spatiaotemporal consistency over time makes these attacks unlikely to succeed as most of them either project static objects or can only maintain consistency for very few number of frames [22]. Additionally, many of the attacks are opportunistic [8]) in that they do not target consecutive frames and would thereby be detected as inconsistent. For example, in displaying images of objects on an LCD monitor mounted on the back of the truck (Sec. 5.5.5), we were creating objects so far as YOLOv3 could determine (even though the monitor was neither large enough nor bright enough to obscure the truck). As the objects were not consistent with their inherent context—i.e., the spatiotemporal attributes of the objects did not match their classes (viz., a stop sign should not be moving)—PercepGuard detected most of the attack instances. For attacks against (multi-) object tracking, where detected bounding boxes' locations are shifted in order to deceive object trajectory predictions [90], existing work used simulated patches to demonstrate feasibility. Because of the static nature of the patch (it must move according to the object it is affixed to), it would be difficult for an attacker to maintain the spatiotemporal consistency of the spoofed object.

Existing single sensing modality, consistency-based defenses, including PercepGuard, are not applicable to object removal attacks (if attacker consistently makes an object disappear), since the features needed for consistency verification do not exist for undetected objects. Leveraging the fusion of inputs from multiple, different sensors [200, 201] can be a potential defense; e.g., using a lidar to cross-check the distance to potential objects perceived by a

camera sensor and assuming that both sensors are not compromised simultaneously [36].

5.6.2 Limitations

False positives: Since PercepGuard adopts a data-driven approach (as supposed to model-based ones where detection rules are pre-defined by domain experts), there are two inherent limitations. First, data-driven approaches are also limited by their training datasets. For example, BDD100K was collected in the US, making it unwise to train PercepGuard on this data set and deploy it in UK where the traffic system is different. We argue that fine-tuning may be sufficient such that we don't have to retrain the model from scratch (Sec. 5.5.3.2). Second, there are corner cases for which false positives may arise, which is why we recommend that an anomaly be issued to the planning module of the vehicle instead of an instruction to ignore the object (Fig. 5.3). To reduce FP and FN, a deeper network trained with a larger dataset can help. Also, we can train different networks for different scenarios, such as residential versus highway because objects with the same class move differently in different scenarios.

Evaluation: Our evaluation of PercepGuard involves small numbers of object categories, e.g., five categories from BDD100K for simulation and three from Carla for both simulation and experiments. We argue that PercepGuard is designed for attack detection instead of object classification, thus from the perspective of safety and security, only coarse-grained categorization is necessary. For example, cars, buses and trucks all belong to "vehicles" because they behave similarly, e.g., they can travel at high speeds which can be dangerous to pedestrians/bikes, another category that is less protected. In any case, PercepGuard is scalable and commercial data sets are likely to be much larger and contain a multitude of classes.

5.6.3 Robustness Enhancement

Operating Points: Currently, we use a simple output comparison rule for attack detection but more sophisticated combination rules can be adopted, such as logic-based methods that consider exogenous information/domain knowledge [38, 217], or soft-decision methods that output a likelihood of an attack instead of a binary decision [200, 201]. In addition, we can

also combine PercepGuard with higher-dimensional consistency checks (e.g., multi-object spatial co-existence) to further enhance the adversarial robustness, similar to [95].

Moving Targets: In this chapter we assume that the adversary possesses white-box knowledge about the context data (e.g., ego-vehicle velocity), thus theoretically it is possible to bypass our defense (with high enough cost or power). In order to prevent such strong attacks, we can randomize the configuration of measurement platform (e.g., perturb camera angle or focal length, ego-vehicle speed, etc.), so that the context becomes a secret (or at least difficult to predict) to an external adversary (similar to [99, 100]). This requires the adversary to know the exact configuration in real-time, which is infeasible.

Post-detection: Currently, we choose to alert the human driver and ask them take over the control. We do not intend to use PercepGuard as a standalone object classifier, nor let PercepGuard interfere with the autonomous decision-making. We may invoke randomizing sensor configuration (e.g., switching to a different camera; see above) once an attack is detected. It might also be possible to integrate PercepGuard's result into perception using a weighted majority-vote framework such as [38] that regards our spatio-temporal result as a necessary condition of the main classification result.

5.7 Chapter Summary

In this chapter, we proposed PercepGuard, which is the first to use spatiotemporal consistency of an object to detect misclassification attacks against the perception module of an autonomous system. We adopt a data-driven approach, which extracts spatiotemporal features from the output of object tracking algorithm and cross-checks the consistency with the class label output by the object detection algorithm. To enhance the adversarial robustness, we incorporate additional context information (such as ego-vehicle velocity). We evaluate PercepGuard against both defense-unaware and defense-aware attacks, using a real-world dataset, a simulated dataset and real experiments. Results show that the false positive rates are as low as 5%, while the average true positive rates are as high as 99% from simulation, and 96% from real-world experiments. PercepGuard can efficiently run in real-time, is agnostic to specific sensing modality, attack methods, and is extensible.

Chapter 6

Impact of Perception Attacks and Robust Prediction and Planning

6.1 Introduction

Autonomous driving systems are generally composed of perception, planning, control, and actuation modules. These modules work together sequentially to make decisions. The perception module comprehends the scene around the ego vehicle using sensors such as cameras and/or lidars, then the machine learning (ML) algorithms analyze the sensor output (e.g., image pixels, lidar cloud points) and produces a high-level understanding of the surrounding, such as where and what a detected obstacle is. The planning module then makes high-level decisions such as decelerating upon this information and pass it to the control module which converts them to low-level commands, e.g., braking, for the actuation module to realize.

Because the perception module is exposed to the outside world, it creates a large attack surface to external attackers who can launch perception attacks. Moreover, because ML models are involved in perception, advanced attackers can exploit the vulnerability of neural networks to guide their perception noise. As we have studied in Chapters 3, 4, and 5, these perception attackers aim for creating a non-existent object [6, 9], removing an existing one [8, 9, 82, 83], altering the object class [21], or hijacking object trackers [90, 218]. To achieve the attack goals, external attackers can modify the physical world (e.g., attaching an LCD monitor onto the back of a vehicle, displaying adversarial noise that can change the vehicle's classification result into a "person") [219], or compromise the sensors themselves (e.g., shining light into the camera to create an adversarial ghost recognized as a "person" [6]).

While extensive effort has been taken into the perception security, it is still unclear how the subsequent modules would react to these attacks (where existing works only evaluate their own attacks with limited scenarios [9, 21, 22]), and more importantly, how to design a

perception attack-resilient planning system. In this chapter, we aim to study the consequences of the aforementioned perception attacks, and correspondingly, develop a robust prediction framework that can tolerate classification uncertainty (while existing work only considers the uncertainty of object dynamics such as their measured size, location and velocity, etc. [220–223]).

There are several challenges involved in the first objective. First, there are many perception attacks with differing attack goals and attack methodologies, which becomes a challenge for us to design an evaluation framework that is general enough to consider all of them without actually having to implement them. Second, while simulating the attack outcomes may be a feasible solution to the first challenge, we must consider the physical realizability of the simulated attacks because otherwise an impractical attack does not bring actual threats to the real world. Building a robust prediction system is also challenging where it should be agnostic to perception attacks and needs to consider adaptive attackers who have the white-box knowledge of the system and aim to deceive it.

In this chapter, we first investigate Baidu's Apollo autonomous driving agent as a case study as it is a state-of-the-art self-driving platform that is also open-source [15, 224]. To address the first challenge, we develop a framework that allows us to intercept the communication between the modules in the decision-making pipeline so that we can arbitrarily modify the messages. For example, we can change the obstacle type, velocity, location, shape, etc. at any time for any given object, therefore we can simulate the outcome of any kind of perception attacks without actually having to implement them (either physically or digitally). We discover that even a small change as simple as altering the object type can produce dangerous actuation on the road. For example, once the preceding car on an adjacent lane is misclassified as a person, the ego vehicle brakes much more frequently than without the attack. This is because modern systems include a prediction module [18, 225], in which future trajectories of obstacles are predicted for precaution purposes. In both platforms, separated procedures are adopted for different object categories due to their naturally distinctive dynamics: Cars tend to run at high speed and follow the lane, while pedestrians are often at low speed and have more freedom. This exposes an attack surface of such autonomous systems the attacker may change the classification result of the preceding vehicle into a person, forcing the ego-vehicle

to decelerate on a highway. Previous works have not jointly considered obstacle classification and their trajectory prediction (hence the planning decisions as well).

Motivated by the above case study, we propose a misclassification attack mitigation method for the prediction module. We take into consideration the uncertainty of classification results, caused either by natural errors or adversarial attacks. We follow the idea of Kalman filters where a dual of prediction and update is iterated. We show that such a greedy strategy is able to minimize the total prediction error. Evaluation results show that, our framework is able to correct 93% of the prediction error caused by misclassification attacks. We integrate our system into Apollo's framework to demonstrate that our framework provides effective resiliency against misclassification attacks.

6.1.1 Chapter Organization

The rest of the chapter is organized as follows. Section 6.2 introduces the background about modern autonomous systems as well as related work. Section 6.3 discusses the system model and the threat model. The hijacking evaluation framework is detailed in Section 6.4, and our defensive solution is presented in Section 6.5. The evaluation setup and results of both are shown in Section 6.6. Finally, we discuss and summarize the chapter in Section 6.7 and Section 6.8, respectively.

6.2 Background

Here we discuss the background about the Apollo platform, especially its cyber channel design. Then, we summarize the common design of the prediction module deployed in modern autonomous driving systems. Finally, we review related work on robust planning and recent advances of trajectory prediction.

6.2.1 Apollo's Architecture

Baidu Apollo is an open-source autonomous driving platform that has been tested on the road. In the perception module, a multi-sensor fusion (MSF) algorithm is used to jointly consider the output from cameras, lidars, and radars. The final results of an obstacle include

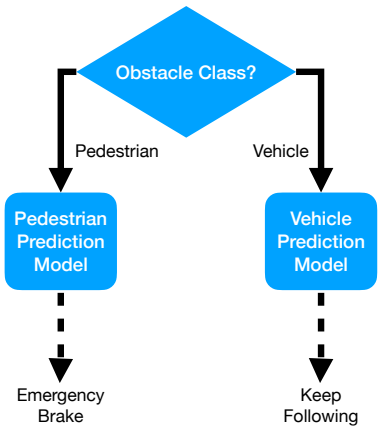


Figure 6.1: Divide and conquer is a common design of prediction modules.

its type, distance to the ego-vehicle, (3D) bounding boxes and velocity estimate. It leverages a prediction module to estimate the future trajectory of nearby obstacles for pre-cautious planning. Deep learning models are widely adopted in Apollo including the perception and prediction modules.

The platform employs the Cyber RT framework [15] for the communication among the modules in the decision-making pipeline. The input and output of a module is directed to a cyber channel, similar to the concept of pipes for a Linux operating system. For example, the perception module reads from `/apollo/sensor/lidar_front` for lidar data and `/apollo/sensor/camera/front_6mm` for camera data, etc. The output channels include `/apollo/perception/obstacles` for obstacles and `/apollo/perception/traffic_light` for traffic lights. Messages are first serialized and then streamed to a channel, containing rich information about an obstacle, such as its type, size and location, etc. In this chapter, we show that the input and output of a module can be re-directed to a different channel (even a new one created by the hijacker), such that the communication among them can be intercepted where we can modify the messages on the fly.

6.2.2 Prediction Module

Modern autonomous driving systems, such as Apollo and Autoware, include a prediction module where future trajectory of obstacles are predicted, so that the planning can be precautious. E.g., a car should decelerate for a pedestrian who is about to cross the street.

In both platforms, separated procedures are adopted for different object categories because their dynamics are distinctive in nature (Figure 6.1), e.g., pedestrians have more freedom than cars do.

In Apollo, a combination of CNN and LSTM models is used to learn trajectory patterns from sequences of semantic maps. More importantly, to achieve higher prediction accuracy, independent CNN+LSTM models are used for different obstacle classes. For instance, the prediction model for vehicles is trained only with vehicle data, and during test time, only those obstacles classified as "vehicles" are fed to the model for prediction. The same criterion applies to pedestrians as well [18]. These two different classes of obstacles naturally behave distinctively; hence the prediction result for each differs as well even though their historical trajectories are very close to each other. This is because these two models are trained with exclusive datasets which contain different trajectory patterns. That is, if we feed the same trajectory to them, different predictions are generated. The planning module then uses these predictions to make decisions on the reactions that the system should take, e.g., to decelerate for the pedestrian who is about to cross the street.

Using the perception attack evaluation platform, we discover that such a common design for prediction exposes a severe vulnerability where an attacker can simply change the object type to jeopardize safety. Therefore, in this chapter we aim to improve the robustness of the current design (Figure 6.1) by considering the classification uncertainty in the loop.

6.2.3 Related Work

Here, we review the related work on decision-making with uncertainty and the prediction algorithms. We also discuss two recent works on the security of trajectory prediction. Recent advance in system-level autonomous vehicle security evaluation is also reviewed.

6.2.3.1 Decision-making with Uncertainty

Due to sensing noise, perception constraints (e.g., limited field of view), and unpredictable human behavior, etc., uncertainty needs to be considered when designing decision-making algorithms. Markov decision process (MDP) has been widely adopted for solving uncertainty in decision-making [221, 222, 226]. Linear-quadratic regulator (LQR)-based methods leverage

Kalman filters to predict future trajectory of both the ego-vehicle and the traffic participants [227]. A Bayesian network (BN)-based framework is proposed to make platooning decisions with uncertain traffic sign recognition results validated by maps or V2X communication [228], which is later replaced with probabilistic convolutional neural networks for the validation [229]. BNs are also used for vehicular trajectory prediction according to maps and other traffic objects [221]. In a model predictive control (MPC) approach, an optimization problem is being solved in an online manner for decisions [220, 230]. Aslansefat et al. propose SafeML frameworks where additional distance metrics such as Kuiper are used for certainty assessment on classification results [231, 232].

These approaches do not differentiate object types in predicting object future trajectory, which is in fact a common practice adopted by modern autonomous driving systems [15, 233]. Therefore, their methodologies can only be applied to the prediction procedures for individual object types which however does not help defend against object misclassification attacks or natural classification error/uncertainty.

6.2.3.2 Trajectory Prediction for Autonomous Driving

To predict future path of traffic agents, many sources of information can be leveraged, such as the historical trajectories, maps, object classes, ego vehicle dynamics plans. Trajectron++ leverages heterogeneous data to make multi-agent behavior prediction where inter-agent interaction is considered [225]. Social LSTM models predict human trajectory with social interaction [234]. GRIP also considers social interaction but averages the predictions from different classes [235]. However, they do not consider the classification uncertainty, and also because of their social interaction consideration, misclassification attacks could make a larger impact when more than one object get misclassified.

6.2.3.3 Prediction Attacks and Defenses

A recent prediction attack [223] where the attacker drives a vehicle to follow the trajectory specially crafted such that the prediction model predicts it is going to change to the ego-vehicle's lane. As a result, the ego vehicle decelerates to avoid the collision. However, this attack requires precise control of the attack vehicle, which is more difficult than launching

misclassification attacks. To counter this attack, an adversarial training-based defense has been proposed by Jiao et al. [236] but it does not work against misclassification attacks because in their threat model they assume the classification result from perception is trustworthy.

6.2.3.4 System-level Security Evaluation

The impact of attacks on autonomous vehicles has only been evaluated individually. Their effects from the system's perspective have recently drawn researchers' attention. Hu et al. [237] propose a platform to consider the recent development of attacks against object detection and traffic light recognition. Wan et al. [238] develop a systematic fuzzing approach to identity denial-of-service vulnerabilities. Instead of simulating individual attacks at a time, in this chapter we propose to jointly study multiple perception attacks and develop a framework that allows us to do so without the implementation in order to relax the computation requirement.

6.3 System Model and Threat Model

We consider an autonomous driving system with a prediction module for future trajectory of obstacles, in which different prediction models, i.e., *predictors*, are used for different object types. Such a divide-and-conquer design is commonly found in modern autonomous systems [15, 233]. The prediction model may be based on neural networks such as recurrent neural networks (RNN), and/or Kalman filters, etc.

We consider attackers who aim to alter the classification results of a target obstacle via perception attacks. The attack can be implemented via physical modification [4, 219], or compromising sensors [6]. We assume the attacker has white-box knowledge of the perception system for crafting adversarial noise, though in evaluation we simulate the attack outcome (i.e., misclassification) using our hijacking framework for Apollo, with realistic considerations such as limited attack success rates.

6.4 Impact of Perception Attacks

We aim to evaluate the impact of perception attacks in an end-to-end manner; that is, we attempt to answer this question: "how do perception attacks affect the planning and

actuation?" The benefits are three-fold. First, it helps us understand the vulnerability of the autonomous driving pipeline from a system perspective, rather than from just a perception sub-module. For example, how would they effect the behavioral and route planning [237–239]? Second, it allows for discovering unknown vulnerabilities. Third, this will enable us to design a defense that can adopt to various types of attacks under different driving scenarios.

6.4.1 Challenges

There are many perception attacks, targeting various sensing modalities and algorithms, therefore it is challenging to efficiently reproduce each of them with limited expertise on that specific setup. Second, when considering a combination of multiple perception attacks, the computational issue raises because most of these attacks require solving a complex optimization problem involving neural networks, not to mention that the autonomous driving agent itself already consumes considerable computation resources. Third, we must consider the attack practicality which is difficult to quantify.

6.4.2 Solutions

We propose to intercept the communication among decision-making modules (Figure 6.2). This way, we can easily modify the messages that are transmitted over the channel; as such, we only need to reproduce the output of a perception attack. For example, to simulate the misclassification attacks [6], we can simply change the `object type` field from the perception obstacle messages. Similarly, for tracker hijacking attacks [218], we can shift the bounding boxes without actually having to solve the optimization problems for the noise.

In this work, we use Baidu Apollo as an example to demonstrate the idea. Suppose we aim to intercept the channel between module A and module B, and we have a hijacking module named C. There are two ways to do so: we can redirect the input stream to module B to the output stream of C, or we may redirect the output stream of A to the input stream of C. Figure 6.2 illustrates the first option. We force module B to read from a channel that C is writing to. To do that, for the prediction module we can edit the file `/apollo/modules/prediction/dag/prediction.dag` and change the `channel: "apollo/perception/obstacles"` into `channel: "apollo/perception/attacked_obsta-`

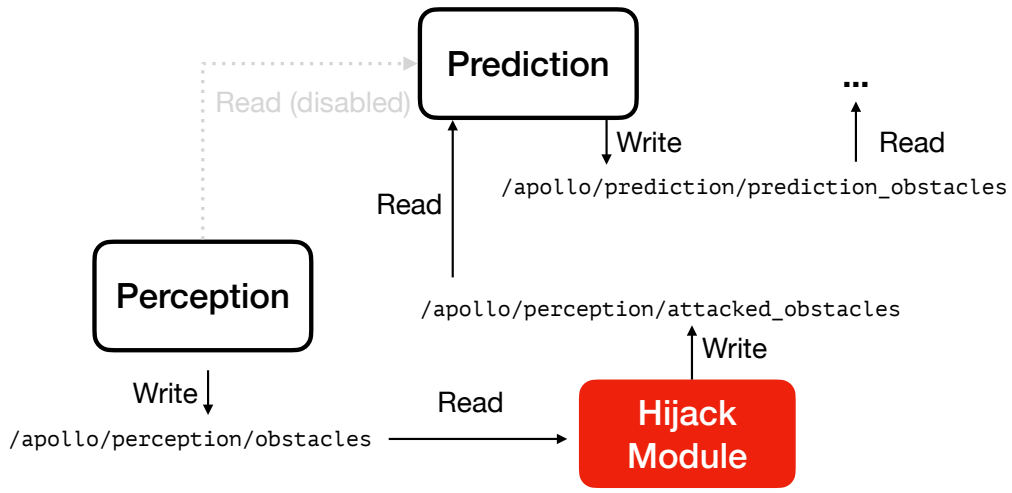


Figure 6.2: Cyber Channel Hijacking

cles". In the hijacking module, we will read from `/apollo/perception/obstacles` and identify the object of interest, and then change its type. Finally, we push the modified message to the channel `apollo/perception/attached_obstacles`.

To quantify the attack practicality, we propose to divide it into attack feasibility and attack cost. The attack feasibility is quantified using the attack success rate where we use the state-of-the-art optimizer and algorithm to solve the attack optimization in which we consider the attack objective, physical constraints (such as the LCD size or the projector's maximum brightness, depending on the attack vectors), as well as defense evasion. The higher the attack success rate is, the more feasible the attack is. Then, we map the resulting, potentially optimal adversarial noise into the physical domain in terms of financial expenses for acquiring and operating the attack vectors.

6.5 Prediction with Class Uncertainty

6.5.1 Problem Statement

In light of various perception attacks/errors that cause uncertainty in decision-making, we aim to develop a defense that corrects the object trajectory prediction under object misclassification attacks. Specifically, a predictor for class c takes the past u trajectory points and predicts v future points, i.e., $\hat{B}_{t+1,t+v} = F_c(B_{t-u,t})$ where $B_{a,b} = \{b_i : a \leq i \leq b\}$ and $b = (x, y)$ from the bird's view. Because the object misclassification attack changes c to c', a

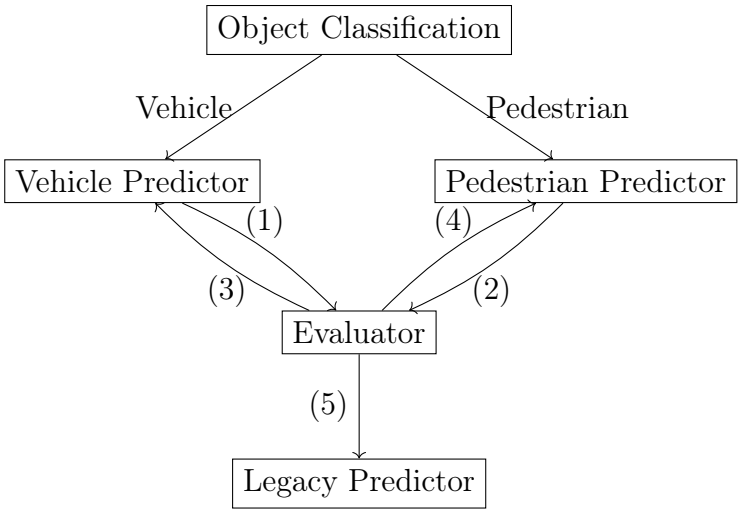


Figure 6.3: We allow switching among predictors when the current one does not fit well. The switching criteria depend on the trajectory similarity and prior information.

different predictor is used: $\hat{B}'_{t+1,t+v} = F_{c'}(B_{t-u,t})$, and if $\hat{B}'_{t+1,t+v}$ deviates from the ground truth $B_{t+1,t+v}$ too much, it may mislead the decision-making. Our objective is to have the predicted trajectory to be as close to the ground truth $B_{t+1,t+v}$ as possible despite the attack.

6.5.2 Solution

Our high-level idea is depicted in Figure 6.3, where If the vehicle predictor's results do not match the recent measurement (1). If the pedestrian predictor's results do not match the recent measurement (2). If the pedestrian predictor was used and actually the vehicle predictor fits better (3). If the vehicle predictor was used and actually the pedestrian predictor fits better (4). If neither of those two fits, we transition into a sub-optimal mode (5) where we use some legacy predictor, e.g., those that are class-agnostic such as the ones that use Kalman filters. The flow chart of the process is plotted in Figure 6.4.

Once long enough trajectory has been recorded, we use the first half of the history to predict the second half: If the prediction matches the ground truth, we say the predictor fits well. Trajectory similarity is defined as $D(B_{a,b}, \hat{B}_{a,b})$ where $D(\cdot)$ is some distance metric. This tells us how accurate previous predictions are.

Assuming the obstacle enters the scenes at $t = t_s$ (Figure 6.5). At t_e we calculate the trajectory similarity $s_e = D(B_{m,e}, \hat{B}_{m,e})$, where $\hat{B}_{m,e}$ is predicted based on $B_{s,m}$. At

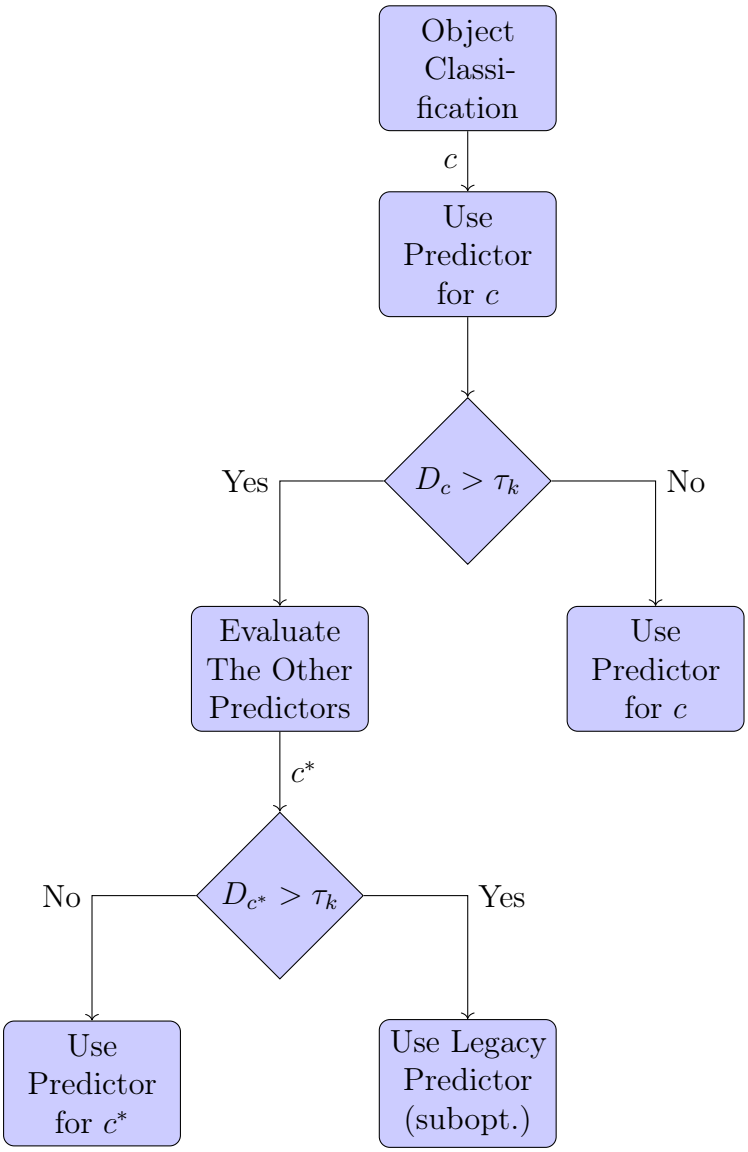


Figure 6.4: A prediction system that considers class uncertainty. The leaf blocks trap, under the assumption that object classes do not change.

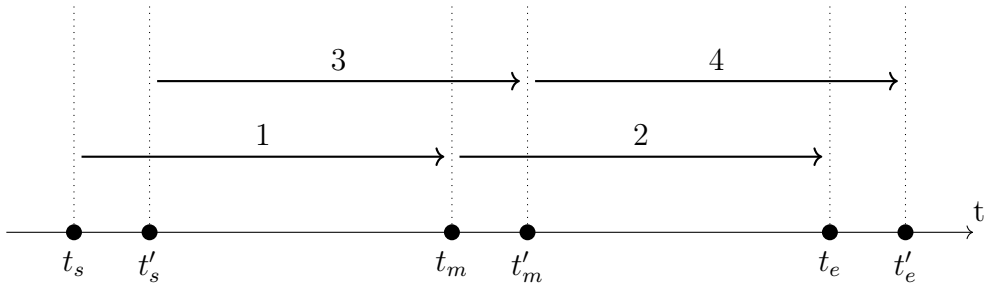


Figure 6.5: Trajectory timeline

t'_e, we calculate $s'_e = D(B_{m',e'}, \hat{B}_{m',e'})$. If, after k steps, the (weighted) average of these similarities/discrepancies exceeds some threshold τ_k, we transition to the evaluator phase because the current predictor does not produce satisfactory accuracy.

The evaluator recalculates the similarities using the other predictors and find one that fits the best. If the best one still exceeds τ_k, we switch to the legacy predictor (e.g., a class-agnostic one), otherwise to the best predictor.

Moreover, we can potentially consider the prior information for reasoning about the object class. The prior is the probability of an object type appears at a given location. For example, a person has little chance of walking on the highway, which can be learned via density estimation. The prior can also be the class confidence from PercepGuard (Chapter 5).

6.6 Evaluation

6.6.1 Misclassification Attack Mitigation

6.6.1.1 Setup

We use Trajectron++ [225], the state-of-the-art trajectory prediction algorithm for autonomous driving. It uses separated LSTM models to learn the dynamics pattern of vehicles and pedestrians, as well as the ego vehicle, in order to predict their future trajectories. It leverages a CNN model to extract information from maps to assist the prediction. Social interaction among traffic participants is also considered by Trajectron++ using attention. We use the pre-trained model published by [223].

We adopt the nuScenes dataset [240] that provides traffic trajectory collected from urban areas. The trajectory is sampled at 2 Hz. It also provides maps. We randomly select 100

scenarios from the dataset and format each of them into 16-slot *segments* where Trajectron++ can observe 4 time slots as trajectory history, and predicts the trajectory for the other 12 time slots. For example, for a segment of 30 time slots $t = [1, 30]$, we generate the first segment from $t = 1$ to $t = 16$, and the second segment from $t = 2$ to $t = 17$, etc. In the first segment $t = [1, 16]$, $t = [1, 4]$ is fed to the prediction algorithm that predicts for $t = [5, 16]$.

The prediction errors are calculated based on the second half of the ground truth. In each segment, there are multiple traffic nodes (such as cars and pedestrians), and one of them is the main node for which Trajectron++ considers the social interaction from the other nodes. To simulate the misclassification attack, we change the class of the main node. Two metrics are considered. The final displacement error (FDE) is the root mean squared error (RMSE) between the final positions of the ground truth and the predicted paths. The average displacement error (ADE) is calculated as the average of the RMSE between the ground truth and prediction (not just for the final position). There are four *cases*:

- Normal (N): There is no misclassification attack, nor our defense deployed.
- Normal+Defense (ND): Non-adversarial case
- Attacked (A): Unprotected adversarial case
- Attacked+Defense (AD): Protected adversarial case

For each segment, we have a loss value (FDE or ADE) for each one of the four cases. We average all segment results of as the result of the scenario.

6.6.1.2 Results

We plot the ADE loss distribution under all four cases in Figure 6.7. Figure 6.7a shows that Trajectron++ performs well under normal circumstances, with loss concentrated around 0. Figure 6.7b tells us that our defense only causes a lightly more loss than the normal case. Figure 6.7c means the misclassification attack is highly effective, resulting large prediction errors. Fortunately, our defense is able to correct the errors significantly (Figure 6.7d).

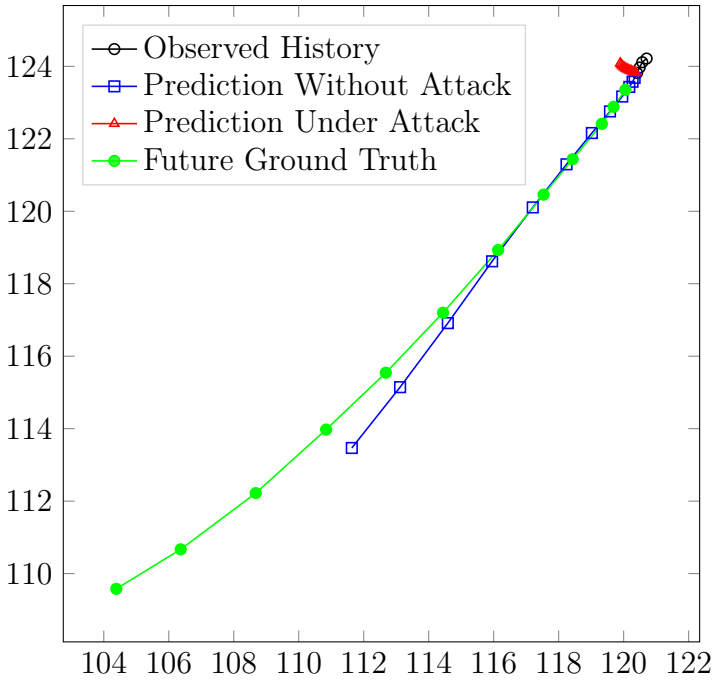
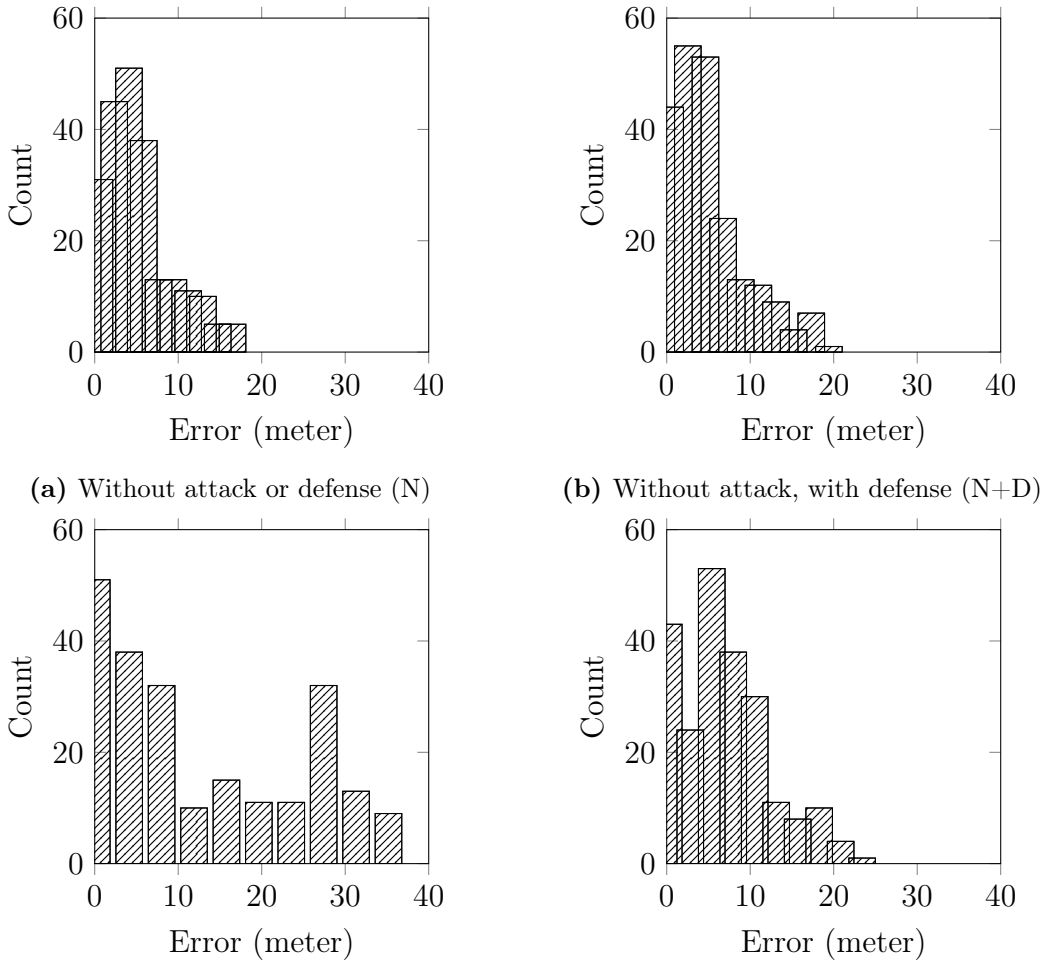


Figure 6.6: Trajectory Example

Lower Attack Success Rate: We also consider a more realistic attack scenario where the attacker cannot induce misclassification constantly, but with a misclassification probability. That is, the probability that the attack is able to succeed the misclassification attack for a frame is lower than one. In this setting, we vary the probability from 1 to 1/4 (Figure 6.8). We can see that the ADE decreases for the "attacked" case; this is because for those unsuccessful frames, the correct trajectory prediction model is used which results in a lower ADE. If we apply our defense to the "attacked" traces, the ADE is largely corrected and becomes close to the "normal" case. Lastly, the ADE of the "normal+defense" case is almost identical to the "normal" case, indicating that our defense does not deteriorate the prediction performance under non-adversarial scenarios, which is similar to the results from Figure 6.7.

Observed Horizon: In addition, we experiment with various history lengths (L_h) that the model can observe for predicting the next L_f-step future horizon. We present both the ADE and FDE in Figure 6.9, which shows the errors first decrease and then increase, resulting in an optimal history length (which is close to the default setting of Trajectron++). Intuitively, the longer history that we can observe, the more accurate prediction we can achieve. We can



(a) Without attack or defense (N)

(b) Without attack, with defense (N+D)

(c) Under attack, without Defense (A)

(d) Under attack, with defense (A+D)

Figure 6.7: Histograms of ADE prediction loss.

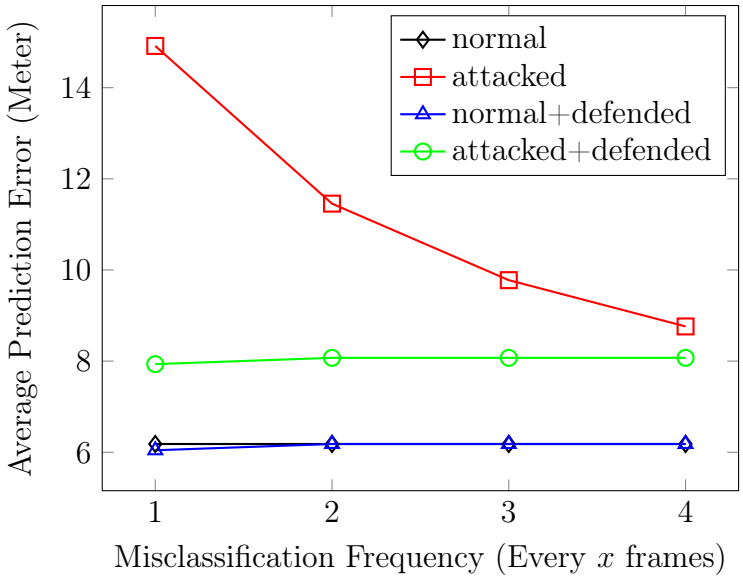


Figure 6.8: Performances with varying attack frequencies.

confirm it from the first part of the curves where the errors decrease as the history length increases. However, for the second part the errors start to increase. Such convex performance is because history from too long ago is not actually helpful and may mislead the prediction. For example, consider a car from the north just finished a "U" turn and is going to the south. If we use the trajectory of the "U" turn for prediction, we may forecast it to keep turning.

6.6.2 End-to-end Evaluation

In this section, we evaluate the efficacy of our hijacking tool and use it demonstrate the end-to-end impact of misclassification attacks, as well as the performance of our defense.

6.6.2.1 Simulation Setup

We run Apollo 6.0 in the LGSVL simulator, where we place an NPC vehicle driving straight forward in front of the ego vehicle on either of the adjacent lanes. The test is conducted using five maps (for highway or urban, etc.). For each map, we repeat the simulation ten times.

6.6.2.2 Attack Impact

From all five maps, similar results are produced: the ego vehicle decelerates to avoid the collision because the obstacle is predicted to change lane. An example is shown in Figure 6.10,

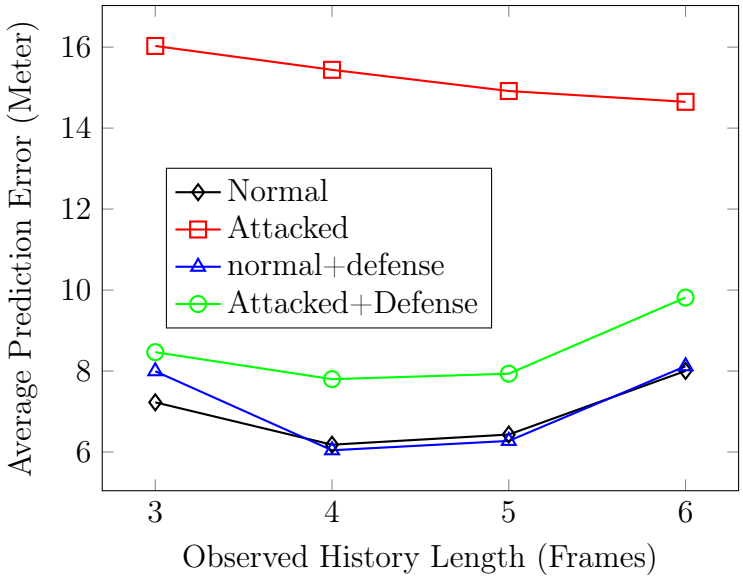


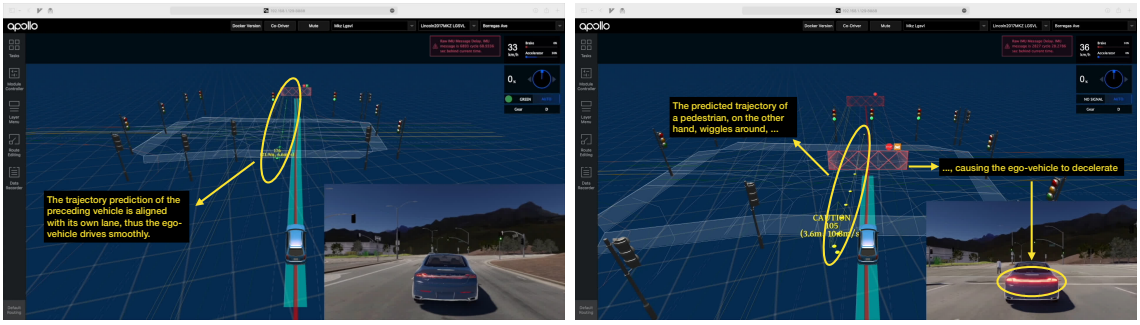
Figure 6.9: Trajectory prediction performance with varying history length (2 Hz).

where due to the attack, the ego vehicle predicts that the object may change to its lane so it performs deceleration. Meanwhile, without the attack the ego vehicle drives smoothly without interruption.

Without the attacks, the ego vehicle accelerates and decelerate smoothly and the speed on the highway is at the speed limit which means it is a safe speed. On the other hand, when under the attack, the ego-vehicle always decelerates on the highway, making it dangerous to the other traffic participants, e.g., a rear collision may occur. This is because it recognizes the attack vehicle as a person so it predicts the "person" may change lane. In local scenarios, the ego-vehicle struggles to accelerate because of the same reason: to avoid hitting the "person", it keeps a distance and speed that it regards as safe, which is incorrect.

6.6.2.3 Defense End-To-End Performance

To investigate the performance of our defense from an end-to-end perspective, we integrate it into the Apollo framework and execute the whole decision-making pipeline in a simulated environment. We run Apollo 6.0 in the LGSVL simulator, where we place an NPC vehicle that is moving forward in front of the ego vehicle on either of the adjacent lanes. We test our defense on both highway and urban maps. For each map, we repeat the simulation for ten times and gather the average result.



(a) Without Attack **(b)** With Attack

Figure 6.10: A dangerous outcome of object misclassification. Because of the attack, the ego vehicle predicts the object may change lane so it performs deceleration, while without the attack the ego vehicle drives smoothly without interruption.

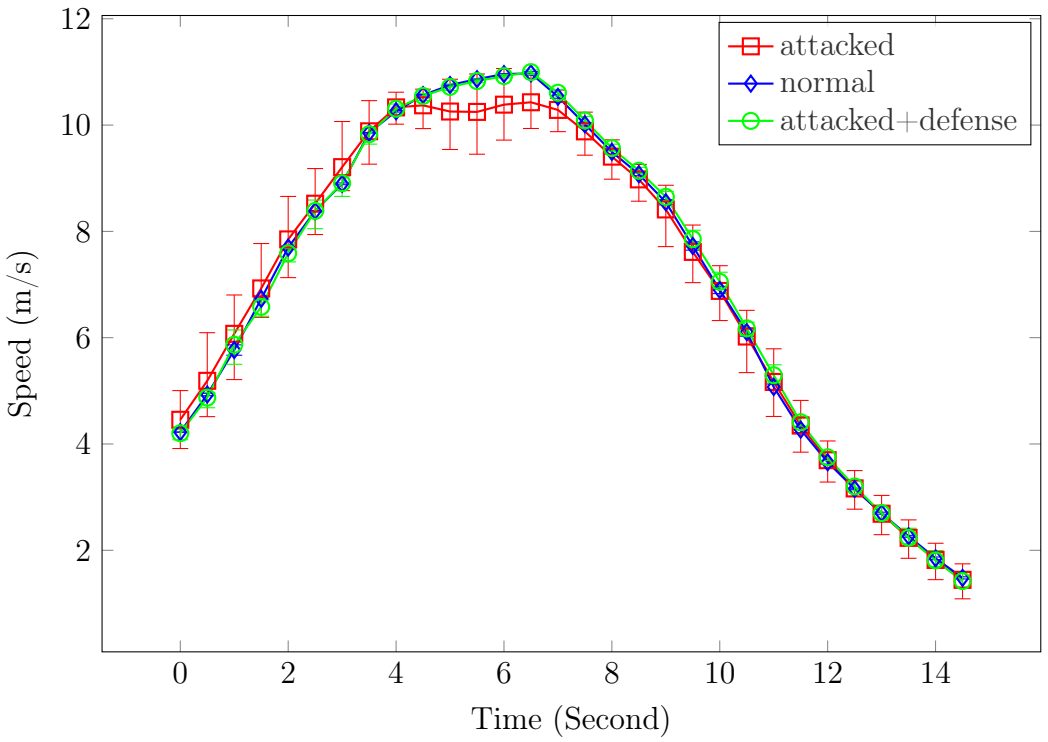


Figure 6.11: Average speed for urban maps

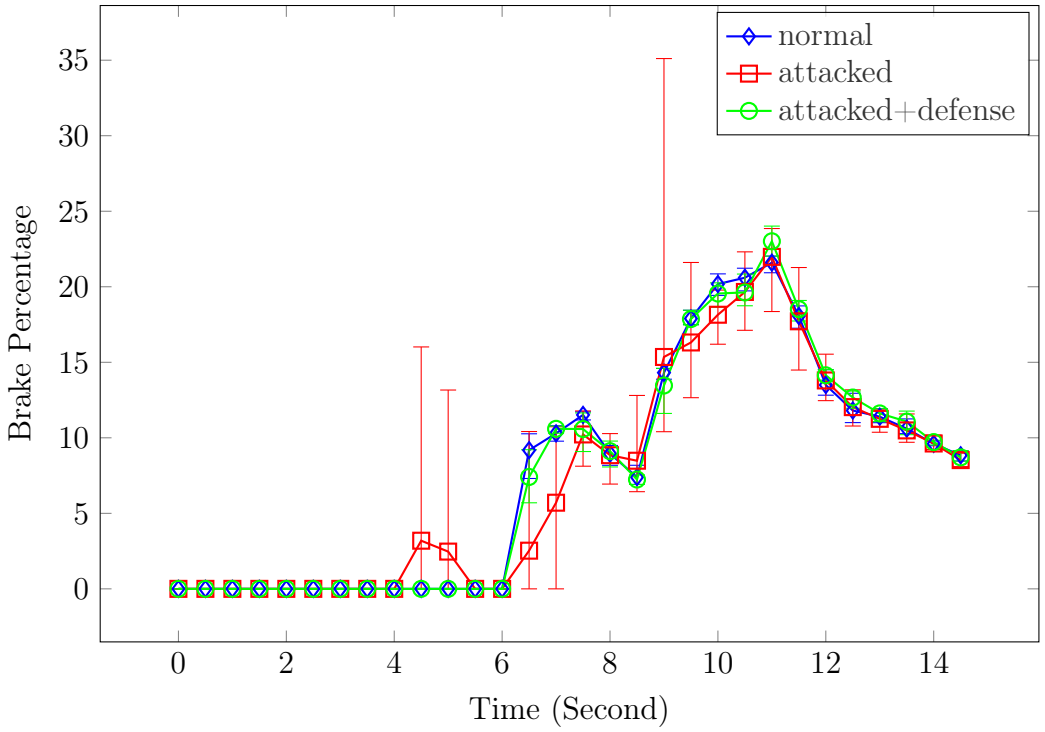


Figure 6.12: Average brake percentage for urban maps

The speed variation and the brake level variation are provided in Figures 6.11 and 6.12, respectively, both in terms of time for the urban map. The error bars show the confidence internals. Under normal scenarios, the brake is not enabled until around the 7-th second when the ego-vehicle is approaching its destination where it prepares to stop. Under the attack, as the ego-vehicle drives past the NPC vehicle that is misclassified as a "person", it brakes frequently because it predicts the NPC is going to change to its lane. This results in unnecessary deceleration and high variation in brake level, which is fuel inefficient and unsafe due to such random traffic patterns. With our defense being deployed, we are able to eliminate such unnecessary braking activities, allowing the ego-vehicle to drive much more smoothly and safely.

Highway scenarios are more critical due to high speed (Figure 6.13). Without the attack, the ego-vehicle quickly accelerates to and then maintains 20 m/s until reaching the destination where it decelerates to 0 m/s. Very similar patterns are seen from all the trials thus the confidence intervals (indicated by the error bars) stay narrow, especially during the cruising period. On the other hand, the misclassification attacks that happen between the 25-th and

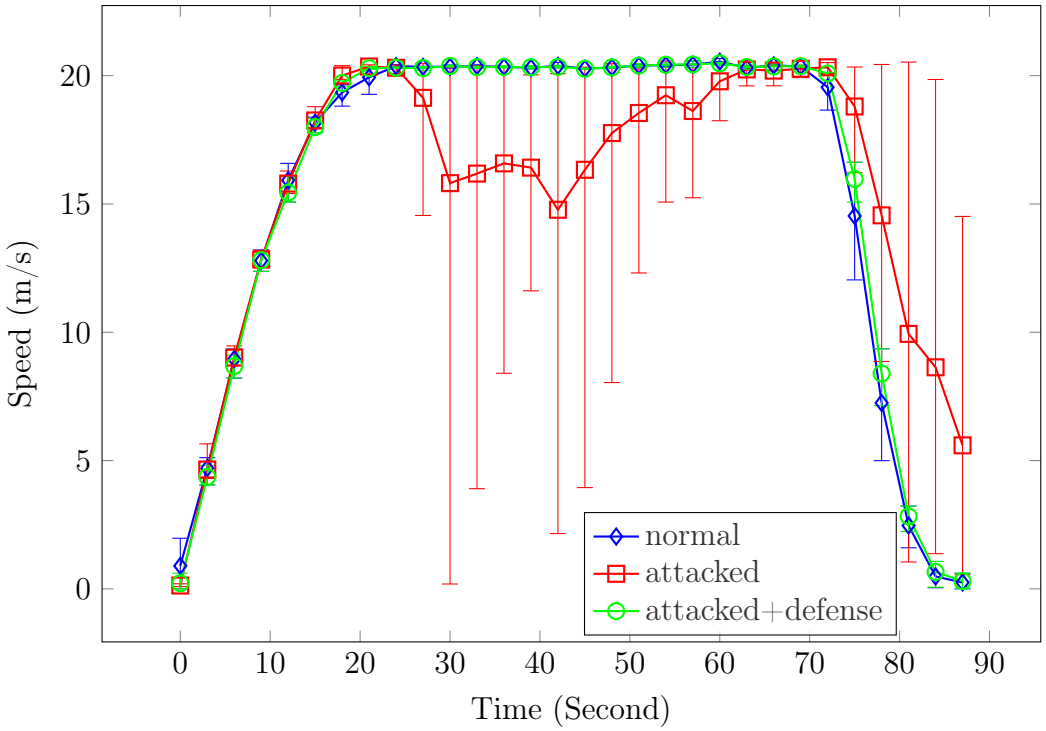


Figure 6.13: Average speed for highway maps

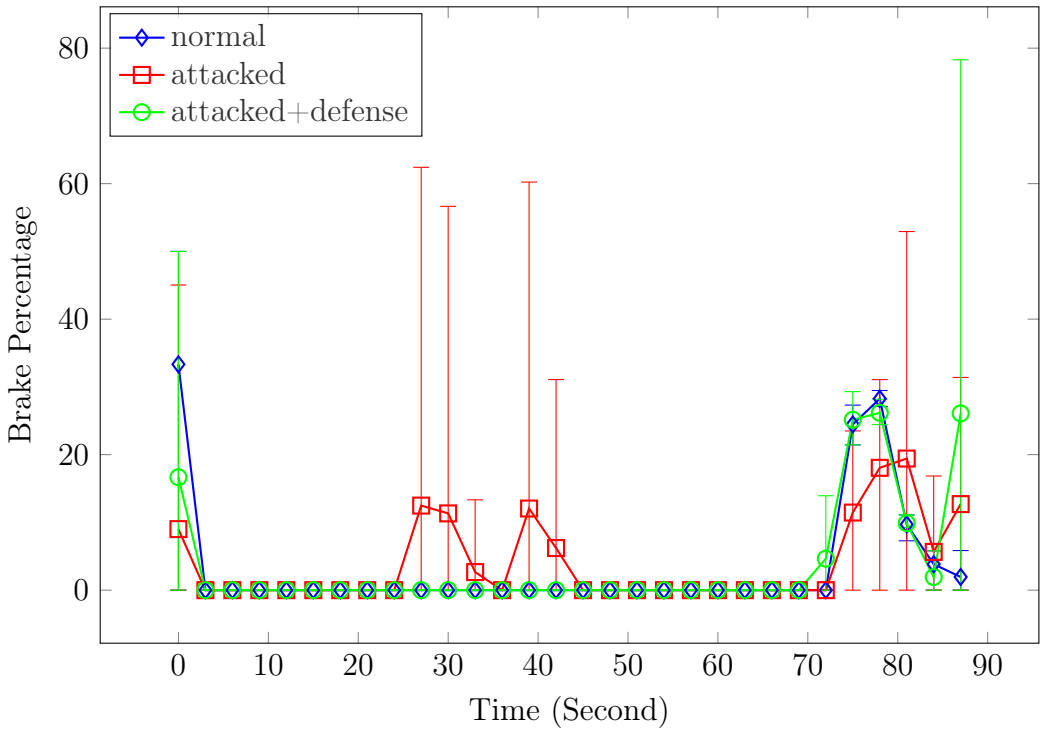


Figure 6.14: Average brake percentage for highway maps.

60-th seconds significantly disrupt the driving pattern, lowering down the speed with frequent brakes. More severely, we observed that the ego-vehicle even *stopped on the highway* (indicated by the error bar between 30-th and 50-th seconds). With our defense being deployed, the speed pattern becomes smooth as normal, and much less deviated. Correspondingly, the brake percentage variation is shown in Figure 6.14, where similar patterns can be seen. The "attacked+defense" case at the last moment has a high brake level because it's approaching the destination but the brake level at the second to the last moment is low.

6.7 Discussion

Lack of History: The default length of observed history by Trajectron++ is 2 seconds, which is sufficient to approximate the momentum of an object. However, for the autonomous driving the obstacle trajectory needs to be forecasted as soon as it enters the scene in order to guide the planning. For instance, a pedestrian obscured by a vision barrier such as a building can suddenly be crossing the street but their past trajectory could not be observed due to the barrier. Though Trajectron++ allows as few as one observed timeslot, its accuracy in that setting is not satisfactory (Figure 6.9). This demands a "one-shot" trajectory prediction, where we aim to predict the trajectory of an obstacle given only one frame.

To this end, one may leverage the pose estimation, prior information, V2V, and maps for such inferences. Pose estimation outputs a finer decomposition of an object rather than just a bounding box, that includes the pixel locations of important body parts [241–243]. For a pedestrian, this includes their head, elbows, knees, etc. The dynamic of pedestrians, such as standing, running, and walking, can be inferred from their poses [241]. In terms of trajectory forecasting, poses can also infer the motion direction and orientation, which allows us to predict their future trajectory given only one frame. Similarly for vehicles, their orientations, wheel angles, brake/reverse lights, etc. can also suggest their future dynamics [243]. On the other hand, prior information such as the trajectory distribution at the point of interest can also be helpful. Maps can further assist one-shot prediction, e.g., the fact that it is a one-way street narrows down the possibilities. Lastly, V2V and/or V2X communication can broadcast the existence and dynamics of traffic participants even before they enter the field of view though the trustworthiness of such messages needs to be carefully evaluated [244–247].

6.8 Chapter Summary

In this chapter, we first investigated the consequences of perception attacks on subsequent modules via hijacking the communication among decision-making modules. We discovered that an attacker can simply change the object classification result to jeopardize safety, fuel economy, and passengers' comfort. Root cause analysis shows that this vulnerability is common in modern systems where separated procedures are adopted for different obstacle classes. Then, we propose a robust prediction system where we incorporate obstacle classification uncertainty. Evaluation with the state-of-the-art prediction algorithm and a real-world dataset shows that, our prediction system is able to correctly predict obstacles' future paths, with much lower prediction errors compared with attacked cases.

Chapter 7

Intrusion Detection Evasion on In-Vehicle Networks

7.1 Introduction

The Controller Area Network (CAN) bus is the de facto industry standard for in-vehicle networks used in modern vehicles for connecting Electronic Control Units (ECUs) that are involved with safety-critical tasks, such as braking and engine control. ECUs are embedded systems with specific automotive-related duty, such as engine control, braking, etc. ECUs are , which require information that is communicated via the CAN bus, consequently the safety of the passengers is directly dependent upon the security of the bus [248, 249]. It has been demonstrated should an attacker gain access to the CAN bus, forged messages can be sent that affect the safe operation of the vehicle (e.g., causing the vehicle to accelerate [250], stopping the engine, disabling the brakes, selectively braking [42], or disabling the transmission [115]). The connection can be through direct connection [42] or via a remotely compromised ECU [43].

Based on existing studies, the confidentiality of CAN messages is not strictly necessary to provide safe operation, whereas authentication and integrity are essential. A CAN frame can only accommodate eight bytes of both data and cryptographic information (Sec. 7.2.1), hence, providing authentication and integrity via message authentication codes (MACs) or digital signatures is not a straightforward proposition. In [45] an out-of-band channel (CAN+ [251]) is leveraged to transmit authentication information; in [252] a delayed authentication scheme is proposed that uses multiple frames to generate a compound MAC; and variable-length MACs are used in [253] to offer protection commensurate with a message's criticality. However, existing solutions to the lack of authentication for CAN messages are either insecure or computationally intensive which makes them incapable of a reliable authentication on a frame-by-frame basis [45]. As strong authentication guarantees cannot be provided for legacy CAN,

intrusion/anomaly detection systems (I/ADS) have been proposed that would at least allow for appropriate countermeasures to be taken in the event of an attack (e.g., ignoring suspect messages or putting the car into a safe state) [118, 120–122, 254, 255] Recently, researchers propose device fingerprinting IDS techniques based on differentiating devices according to either timing [24] or the physical-layer characteristics of frames, such as voltage [23, 48]. However, most approaches rely on multiple frames sent by an ECU to evaluate and update the fingerprints.

In this thrust, we first demonstrate that state-of-the-art device fingerprinting-based intrusion detection and identification systems (e.g., [23, 24]) are fundamentally vulnerable to *Hill-climbing-style* attacks, because they either entirely or partially depend on multiple frames to identify the sender of a frame. Due to the multi-frame dependence, a Hill-climbing-style attacker is able to exploit a compromised ECU to impersonate another ECU without being detected *or* identified. The attack is achieved by carefully injecting an increasing amount of malicious messages among legitimate messages, so as to gradually shift the profile of the target ECU toward the attacker's. Since during consecutive time periods the profile only shifts slightly, the attack remains undetected or unidentified. Ultimately, the attacker ECU will be able to inject a majority, or replace all, of the frames sent by the target ECU with its own. Our attack can be regarded as a type of online data poisoning attack against machine learning systems (especially, for classification) which acquire/update training data in an online manner.

Motivated by this attack, we propose SIMPLE, a SIngle-fraMe based Physical-LayEr identification solution to detect intrusion and identify the source of each single CAN frame that is transmitted on a bus by a specific ECU regardless of its message ID. We extract unique voltage-based features (*fingerprints*) in the time-domain from each individual CAN frame transmitted and contribute to an ECU. Unlike existing multi-frame IDS systems, SIMPLE performs secure updates of training data by modelling and compensating for the changes in environment and operating conditions (e.g., temperature and supply voltage). Since fingerprinting in SIMPLE is done on a *per-frame* basis and is very computationally lightweight, the detection can finish even before a frame's transmission ends, thus enabling real-time prevention of intrusion attacks by invalidating the spurious frame before it takes

effect on the vehicle. SIMPLE is a single-frame based intrusion detection and identification system that (to the best of our knowledge) for the first time achieves attack prevention with secure updates.

Section 7.2 provides a brief background on the preliminary of the CAN protocol, the applications of PLI, and a brief survey of existing works. Sec. 7.3 presents our attack model. Sec. 7.4 demonstrates the Hill-climbing-style attack against two state-of-the-art fingerprinting schemes. Sec. 7.5 presents the evaluation and its results. Finally, we draw our conclusion in Sec. 7.7.

7.2 Background

In this section, a brief overview of CAN protocols, followed by basics of physical layer identification are given at first. Later we survey the related works and discuss the basic assumptions of our system.

7.2.1 CAN protocol

CAN protocol was created in 1986 by Robert Bosch GmbH [114], providing broadcast-based communication among ECUs within in-vehicle networks. It is a two-wire, half-duplex bus generating CAN High (CANH) and CAN Low (CANL) signals as output. The details of the CAN protocol is given in [256]. All the ECUs are connected to the same bus, thus they can receive all the frames broadcasted on the bus. There are a few features of the CAN protocol that are related to this work. First, each ECU can be assigned one or multiple message *identifiers* (IDs) that it can send out, which usually represents the data type. Two or more ECUs that want to transmit at the same time have to participate in so-called *priority-based arbitration* in order to occupy the CAN bus. The lower the numerical value of the ID is, the higher the priority the message has. Second, any ECU that observes an error in a frame will transmit an Error Frame that will cause other ECUs to discard the previous/current frame (an ACK slot at the end of each CAN frame allows the ECU that transmitted the frame to determine if a single ECU successfully received the frame) [114]. In addition, message reception is not based on destination address but on message ID (for example, an engine ECU

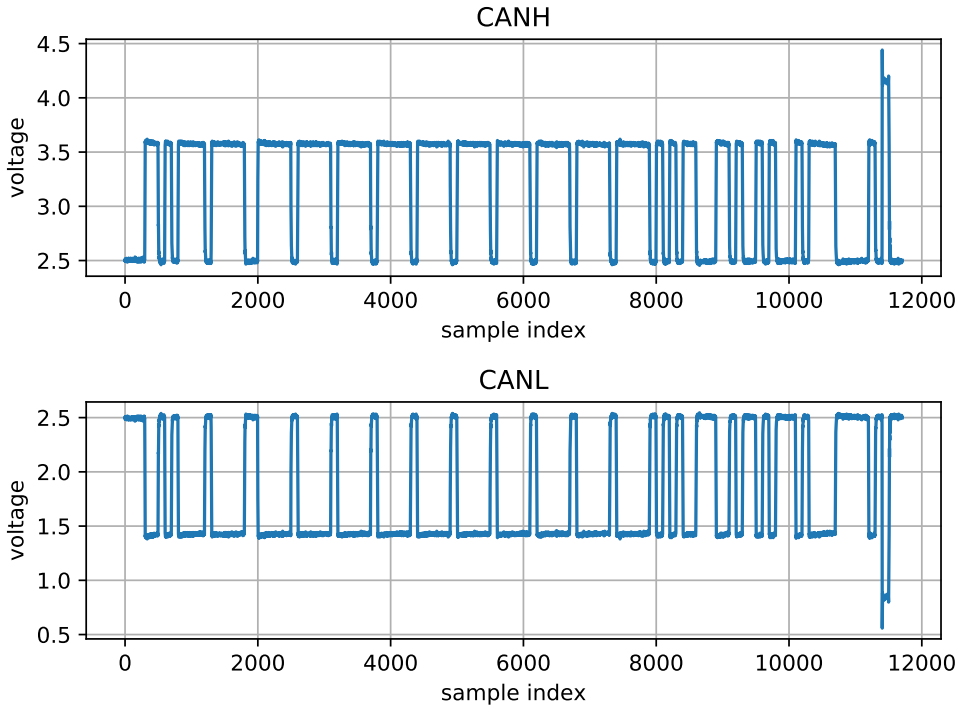


Figure 7.1: A CAN frame captured from a Nissan Sentra.

is programmed to receive only certain subset of "interesting" message IDs related to engine status/control). From the security aspect, weak integrity check is performed by calculating the cyclic redundancy checks (CRC) in each frame. Considering that an ECU can forge the ID, the protocol lacks strong authenticity, which can be fulfilled by using physical layer identification. In a nutshell, several features of this protocol make it unique. For example, arbitrary identification numbers (IDs) instead of addresses are assigned to the CAN frames. This feature results in the lack of source identification in the messages crafted based on this protocol on the one hand, and adds a capability called *priority based arbitration* to it, on the other hand. The priority based arbitration is used for solving the contention problem in the network, where an ECU holding the lowest numerical value of the message ID will have the highest priority to transmit on the bus. To attain this capability, every CAN message is labelled by a unique identifier that is 11 or 29-bit depending on the type of protocol in use. Another specification of CAN is having open drain transceiver drivers, where all of the units are able to drop CAN frames simply by changing the state of the ACK slot at the end of each frame. In fact, the communication in this protocol has a broadcast nature, hence, all

the ECUs connected to the bus can receive the message, and need to perform an acceptance test by matching the content of the identifier to the tasks that each ECU is involved with because CAN messages do not carry an address of the destination.

Physical Layer Identification (PLI) takes into account the hardware and manufacturing inconsistencies that cause minute and unique variations in the signalling behaviour of devices, and translates them into features that can provide reliable identification. A typical PLI system includes three major steps, namely, data acquisition, feature extraction and decision [117]. The data acquisition converts an analogue voltage to a digital signal using an analogue to digital converter (ADC) for further processing; the feature extraction module performs the task of acquiring *fingerprints* for the devices by leveraging the statistical or physical layer characteristics of the signals collected in the previous step and using them to find features; and lastly, the decision module compares the extracted fingerprints from the training data with the ones from the test data using a specific distance metric and defines a threshold for final identification decision making based on how close the features are. Interested readers are encouraged to read about PLI and its applications in [257, 258].

Our proposed scheme SIMPLE also leverages PLI. Among the diverse techniques that are available, we use the Fisher Discriminant Analysis (FDA) transform, which is a dimension reduction technique, followed by a distance calculator named Mahalanobis distance [259], so as to extract and process relevant features from the analog signals and use them as the discriminators in our system.

Existing PLI systems can be categorized into *timing-based* methods [24] and *voltage-based* methods [23, 48]. Existing works (e.g., [23], [24], etc.) in both timing and voltage-based categories rely on multiple messages to make detection and identification decisions. We found that, such reliance on multiple messages makes them vulnerable to a variance of Hill-climbing-style attack [260], in which the adversary is able to inject carefully chosen fraction of malicious messages without being either detected or identified. Furthermore, the adversary is able to iteratively increase the injection rate so that the detection and identification decision threshold can be shifted. We will demonstrate such vulnerability later in Sec. 7.4. Even though VoltageIDS and Scission does not rely on multiple messages, it is not motivated by the vulnerability of multi-frame-based IDS techniques. Besides, they

exploit features in both time and frequency domain, which leads to high complexity.

Most existing works (e.g., [23, 24] etc.) in both timing and voltage-based categories rely on multiple messages to make detection and identification decisions. We found that, such reliance on multiple messages makes them vulnerable to a variance of Hill-climbing-style attack [42], in which the adversary is able to inject carefully chosen fraction of malicious messages without being either detected or identified. Furthermore, the adversary is able to iteratively increase the injection rate so that the detection and identification decision threshold can be shifted. We will demonstrate such vulnerability later in Sec. 4. Even though VoltageIDS [48] and Scission [44] does not rely on multiple messages, it is not motivated by the vulnerability of multi-frame-based IDS techniques. Besides, they exploit features in both time and frequency domain, which leads to high complexity.

Motivated by the vulnerability of multi-frame-based IDS, we propose voltage based PLI scheme, SIMPLE [47], that chooses features only from the time domain and avoids the complexity of frequency domain transformations. It performs a single-frame based detection that is not vulnerable to changes in ambient conditions. Hence, we avoid the unnecessary feature retraining every time the vehicle gets restarted. Both are accomplished by using a higher resolution and higher sampling rate sampler; i.e., a cost-vs-security tradeoff is made to detect an attack using a single frame. Specifically, while Viden [23] only needs to acquire a few samples per frame over multiple frames, and thus needs only a low-rate sampler, SIMPLE needs multiple samples for a single frame and thus requires a high-rate sampler. Nevertheless, SIMPLE is still practical since a sufficiently high resolution and high sample-rate sampler can be acquired for less than $10 per CAN bus [261]. Additionally, unlike Viden, we do not require a separate intrusion detection system (IDS) next to our identification system. Unlike CIDS [24], we can handle both periodic and aperiodic messages. Finally, our approach incurs lower overhead and cost than VoltageIDS [48], as we require fewer samples and a lower sampling rate.

All the voltage-based identifiers given in Table 7.1 illustrate a non-zero value for the

[1]Scission used a slightly different setup for the prototype in which each ECU has a different stub length (2.45 - 13 meters) necessitating stub terminations. This imperfection in CAN bus topology can effect the voltage profile of the ECUs and bias the results knowing that the strongest features in Scission are extracted from the overshoot at the rising edge. In the setup used by SIMPLE however, the length of stubs are all identical and short enough (less than 5 cm) to be negligible.

Table 7.1: Comparison among voltage-based approaches in sampling rate (S.R.), false negative (F.N.), true possitive (T.P.), time complexity (T.C.), signal type (S.T.), environmental compensation (E.C.), secure feature update (S.F.U.), unknow ECU (U.E.).

	Viden [23]	VoltageIDS [48]	Scission[1] [44]	SIMPLE in lab	SIMPLE in vehicle
S.R.	50 KS/s	2.5 GS/s	20 MS/s	500 KS/s	1 MS/s
F.N.	0.2%	3.52%	0.15%	0%	0.899%
T.P.	99.8%	96.48%	99.85%	100%	99.1%
S.T.	CANH&L	Diff.	Diff.	CANH&L	Diff.
T.C.	$\Omega(n^2)$	$\Omega(n \log n)$	$\Omega(n \log n)$	$\Theta(n)$	$\Theta(n)$
E.C.	No	No	No	Yes	Yes
S.F.U.	No	No	Yes	Yes	Yes
U.E.	No	No	No	Yes	Yes

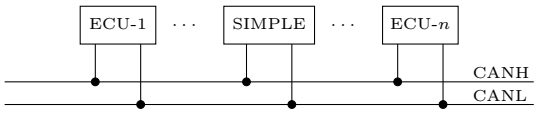


Figure 7.2: SIMPLE runs on a device which is placed on the CAN bus just as other normal ECUs.

error rates. As small as these values are, not even a frame can go wrong for safety critical applications such as air bags. Further efforts like augmented solutions need to be taken to provide the required security for such applications. SIMPLE has the lowest time complexity (Table 7.1), while Viden takes $\Omega(n^2)$ time for each update due to the use of the Recursive Least Squares algorithm. VoltageIDS and Scission need $\Omega(n \log n)$ time because they perform Fourier Transforms in every update to obtain frequency domain features.

7.2.2 Preliminaries of Viden and CIDS

In this section, we will re-state the details of CIDS and Viden.

7.2.2.1 CIDS

In order to detect the intrusion and identify the attacker ECU, Cho et al. proposed Clock-based IDS (CIDS) where the clock skews are used as the fingerprints of ECUs. This is based on the fact that the clock in every ECU advances differently. The clock skew is defined as the difference between the advancing rate of the estimated clock and the true clock. For example, after t seconds, a clock reports the elapsed time as t' seconds. The skew of this clock is then $\frac{t'-t}{t}$. However, since a CAN frame does not contain a timestamp, CIDS

updates the clock skew by evaluating the arrival timestamps of a batch of n messages, i.e., the moments when these messages arrive at the receiver.

Specifically, at k-th step (during $t[k-1]$ to $t[k]$), n arrival timestamps (a_i, for $n = 1 \dots n$) are recorded. The interval between i-th and $i-1$-th arrival timestamps is $T_i = a_i - a_{i-1}$. CIDS calculates the upper and lower control limits $L^+[k]$ and $L^-[k]$ as follows:

$$\mu_T[k] \leftarrow \frac{1}{n}\sum_{i=1}^{n} T_i, \tag{7.1a}$$

$$O[k] \leftarrow \frac{1}{n-1}\sum_{i=2}^{n} a_i - (a_1 + (i-1)\mu_T[k-1]), \tag{7.1b}$$

$$O_{acc}[k] \leftarrow O_{acc}[k-1] + |O[k]|, \tag{7.1c}$$

$$e[k] \leftarrow O_{acc}[k] - S[k-1]t[k], \tag{7.1d}$$

$$\mu_e[k] \leftarrow \frac{1}{k}\sum_{i=1}^{k} e[i], \tag{7.1e}$$

$$\sigma_e^2[k] \leftarrow \frac{1}{k}\sum_{i=1}^{k} (e[i] - \mu_e[k])^2, \tag{7.1f}$$

$$L^+[k] \leftarrow \max\left[0, L^+[k-1] + \frac{e[k] - \mu_e[k]}{\sigma_e[k]} - \kappa\right], \tag{7.1g}$$

$$L^-[k] \leftarrow \max\left[0, L^-[k-1] - \frac{e[k] - \mu_e[k]}{\sigma_e[k]} - \kappa\right]. \tag{7.1h}$$

If either of the control limits $L^+[k]$ or $L^-[k]$ exceeds $\Gamma_L = 5$, CIDS declares a detection of intrusion. If the adversary wants to defeat the intrusion detection, it can simply add the following constraints to Equation 7.11:

$$L^+[k] < \Gamma_L, \tag{7.2a}$$

$$L^-[k] < \Gamma_L. \tag{7.2b}$$

With the *accumulated clock offset* O_{acc}, the *identification error* $e[k]$ and the elapsed time

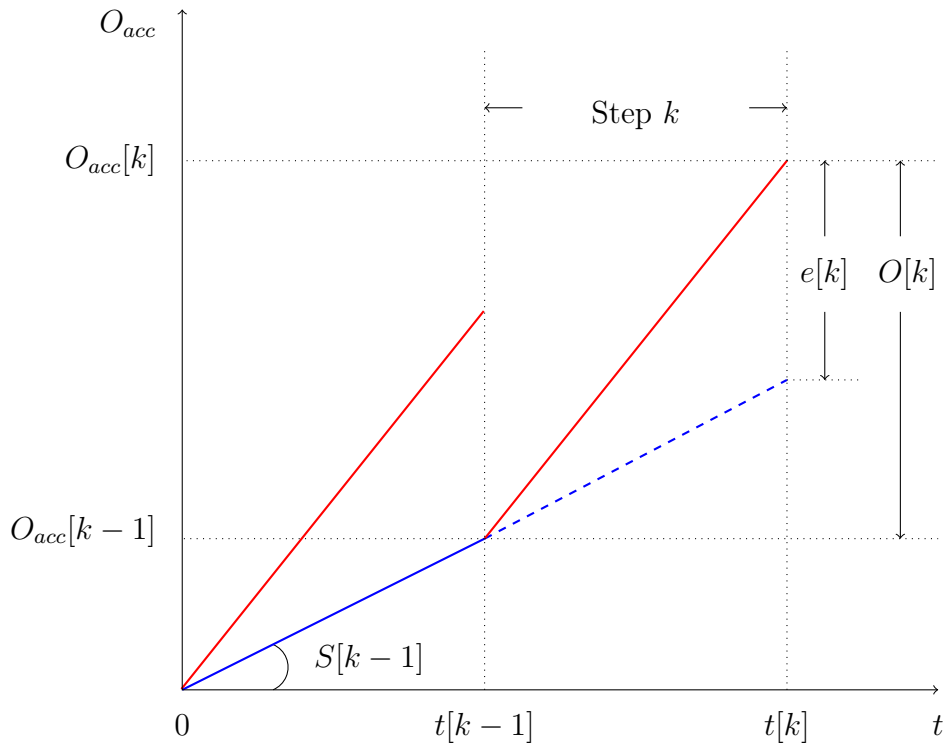


Figure 7.3: Accumulated clock offsets O_{acc}. From time 0 to $t[k-1]$, ECU $\mathbb{A}$ sends messages $m_{\mathbb{A}}$. Its O_{acc} is plotted in red solid line. Meanwhile, ECU $\mathbb{B}$ sends messages $m_{\mathbb{B}}$, plotted in blue solid line. From $t[k-1]$ to $t[k]$, the adversary mounts a masquerade attack, where $\mathbb{B}$ is suspended and $\mathbb{A}$ is programmed to send $m_{\mathbb{B}}$ instead. $m_{\mathbb{B}}$'s new O_{acc} is plotted in red solid line, which is different from the norm clock behavior (the blue dash line). The identification error $e[k]$ indicates how far the accumulated clock offset deviates from CIDS's expectation at $t[k]$. Based on $e[k]$, CIDS decides whether the intrusion exists or not. Furthermore, the slopes of two red solid segments being similar, provides the identification information. In other words, this tells CIDS that the attack messages are sent by ECU $\mathbb{A}$.

$t[k]$, a linear parameter identification problem can be formulated as:

$$O_{acc}[k] = S[k] \cdot t[k] + e[k], \tag{7.3}$$

where the *clock skew* $S[k]$ (i.e., the slope) can be learnt using the Recursive Least Squares (RLS) algorithm. The slope $S[k]$ is viewed as the *norm clock behavior* that CIDS uses to determine the expected timing behavior of the legitimate ECU. See Figure 7.3 for an illustration of these terms.

Algorithm 3: Update the voltage profile Υ at step k

1 **function** `UpdateDispersion`(V, Λ, P^*)**:**
2 | **return** $\Lambda \leftarrow \Lambda + \alpha(P^* - \frac{\#(V<\Lambda)}{\#V})^3$
3 **while** *# of dominant voltage samples collected* $\geq \kappa$ **do**
4 | $V_H, V_L \leftarrow$ past κR CANH and CANL measurements
5 | $v_1 \leftarrow$ the most frequently measured CANH voltages from κ values
6 | $v_2 \leftarrow$ the most frequently measured CANL voltages from κ values
7 | $v_3 \leftarrow$ `UpdateDispersion`(V_H, v_3, 0.75)
8 | $v_4 \leftarrow$ `UpdateDispersion`(V_L, v_4, 0.25)
9 | $v_5 \leftarrow$ `UpdateDispersion`(V_H, v_5, 0.9)
10 | $v_6 \leftarrow$ `UpdateDispersion`(V_L, v_6, 0.1)
11 | **for** $x = 1 \ldots 6$ **do**
12 | | $CVD_x[k] = CVD_x[k-1] + \Delta[k](1 - \frac{v_x[k]}{v_x^*})$
13 | $\Psi[k] = \sum_{x=1}^{6} CVD_x[k]$
14 | $\Psi_{accum}[k] = \sum_{i=1}^{k} \Psi[i]$
15 | $\Upsilon \leftarrow$ The slope of Ψ_{accum} with respect to t learnt by the RLS algorithm

7.2.2.2 Viden

Since CIDS can deal with only the case when the messages are sent periodically, Cho et al. proposed the Voltage-based attacker identification (Viden) [23] that learns the voltage output of the transmitters as the fingerprints of them. Cho et al. assumed there is a perfect underlying IDS that detects intrusions in the first place. Once an intrusion is detected, the suspicious messages are submitted to CIDS [24]. CIDS tries to identify the source of these messages. If CIDS fails, these messages are then submitted to the voltage-based identification model.

Viden is detailed in Algorithm 3. Whenever κ dominant voltage values are sampled, Viden derives the *voltage instance* $v_{1\ldots6}$ from the latest κR samples (Line 5 through Line 10), representing the momentary voltage output character. Then, Viden uses the latest voltage instance to update the *voltage profile* Υ (Line 11 through Line 15) and regards it as the fingerprint of the ECU. Once the underlying IDS detects an intrusion and submits a batch of suspicious messages, Viden runs the same algorithm to compute the *suspicious voltage profile* Υ' and matches it with the voltage profiles that Viden has learnt and trusts. Whichever the closest is decided as the attacker.

More details of Algorithm 3 are given as follows. $v^*_{\{1,3,5\}} = 3.5V$ and $v^*_{\{2,4,6\}} = 1.5V$. $CVD_{\{1\ldots6\}}[0] = 0$. $\alpha = 0.5$. $\Delta[k]$ is the elapsed time since step $k-1$.

When voltage samples from two ECUs are mixed, the mixed profile can be approximated as the linear combination of the two profiles under the assumption that the variants of voltage outputs from all ECUs are close. Specifically, let us suppose ECU $\mathbb{A}$'s V_H distribution follows $\mathcal{N}(\mu_{\mathbb{A}}, \sigma^2)$ and $\mathbb{B}$ follows $\mathcal{N}(\mu_{\mathbb{B}}, \sigma^2)$. In the mixed samples, r of them are from $\mathbb{A}$ and the rest are from $\mathbb{B}$, then the mixed samples' distribution is $\mathcal{N}(r\mu_{\mathbb{A}} + (1-r)\mu_{\mathbb{B}}, \sigma^2)$. The feature v_i (for $i = 1..6$) of the mixed samples can be calculated as:

$$\begin{aligned} v_i &= [0.68 - \mu]/\sigma \\ &= [0.68(1 + r - r) - (r\mu_{\mathbb{A}} + (1-r)\mu_B)]/\sigma \\ &= [r(0.68 - \mu_{\mathbb{A}}) + (1-r)(0.68 - \mu_{\mathbb{B}})]/\sigma \\ &= r v_i^{\mathbb{A}} + (1-r) v_i^{\mathbb{B}}. \end{aligned} \tag{7.4}$$

Similar derivations can be applied to all the other features, including v_1 and v_2 because they are basically the $50th$ percentiles for CANH and CANL, respectively.

Viden also uses a machine classifier based on random forest to defeat against the time-voltage-aware adversary who attempts to tune its voltage output to mimic the legitimate ECU. Since the weak adversary we assumed in the Hill-climbing attack does not attempt to do so, the machine classifier will not be triggered. As a result, we omit the details of the classifier here.

Viden's reliance on multiple messages can be seen by the fact that deriving a voltage instance needs the latest κR voltage samples, which means the minimal number of messages required is

$$\begin{aligned} n &= \frac{\#\text{ of voltage samples needed } (\kappa \cdot R)}{\frac{(\%\text{ of dominant bit}) \times (\text{CAN frame max size})}{\text{transmission rate}} \times (\text{sample rate})} \\ &= \frac{15 \cdot 10 \text{ Samples}}{\frac{50\% \cdot 108 \text{ bit/msg}}{500 \text{ Kbps}} \cdot 50 \text{ Ksamples/sec}} \approx 28 \text{ messages.} \end{aligned} \tag{7.5}$$

In [23], the authors claimed 2 to 3 messages would be enough to derive a voltage instance. That is because they did not consider the $R = 10$.

7.3 Attack model

We consider an adversary capable of compromising one or more ECUs either remotely via wireless interfaces (e.g., the telematic port [115]) or physically (e.g., via OBD-II [42]). Once compromised, the ECU is under full control of the adversary and becomes an *attacker ECU*. With the full control, the adversary can either suspend the ECU or even re-program it to inject arbitrary messages, e.g., use arbitrary message IDs to impersonate another ECU and/or transmit messages containing forged/spurious data. These messages are called *attack messages*. For example, the attacker could compromise an ECU belonging to the entertainment system, and send out "accelerate" or "shut down engine" commands under a different message ID (which is normally sent by the engine control ECU), so as to spoof the engine to carry out wrong actions. We assume that the adversary is aware of the IDS that is installed on the CAN bus and the adversary is able to implement the same algorithms as the IDS. The adversary can also obtain necessary information that can be measured on the CAN bus (e.g., timing and voltage information of other ECUs) using the compromised ECU. Note that this is also assumed by CIDS [24] and Viden [23]. We additionally assume that the ECUs are equipped with tamper-proof temperature sensors.

Furthermore, there are two types of objectives that the adversary may wish to achieve for an impersonation attack. First, it may choose to pursue *dominant impersonation* where at the end a majority of the messages with the targeted message ID are attack messages; or it can choose to pursue *complete impersonation* which is even stronger as the adversary is now able to replace *all* legitimate messages with attack messages. The methods to achieve these attacks will be detailed in the next section. We assume that the attacker can either directly inject its own frame onto the CAN (when no other ECUs are transmitting) or preventing another ECU from transmitting. The latter can be done by synchronizing to the regular messages on the CAN bus and play attack messages right before the one that a legitimate ECU is about to send, or by contending with the latter during the arbitration phase [114, 116].

Note that, the attacker ECU could also inject false data under its own ID, however this is not as effective as impersonation, because CAN messages are addressed by its message IDs which typically represent the data type (e.g., an engine ECU would ignore a message

with one of the IDs belonging to an entertainment ECU). Hence, false data injection attack detection is out of scope of this work. Detection of denial-of-service attacks, such as the bus-off attack [116] will also be studied in the future.

In Section 7.4, we will defeat the multi-frame based fingerprinting algorithms (CIDS [24] and Viden [23]) which also target the impersonation attack as we described above using PLI. Thereby we assume only one ECU is compromised. Readers may be familiar with the Masquerade attack [24], where two ECUs are compromised at the same time, one of which is suspended and the other one is programmed to send attack messages (see Figure 7.1). Compared with the adversary in the Masquerade attack, our attacker requires weaker assumptions since ours only needs to compromise a single ECU.

7.4 Vulnerability of Multi-Frame based Fingerprinting Systems

In a multi-frame based fingerprinting system, a batch of multiple frames has to be collected in order to perform one update of the fingerprinting record/threshold. Such fingerprinting schemes are vulnerable to the so-called *Hill-climbing-style* attack, where the adversary is able to control the quantity of attack frames among the batch of frames collected, so that the attacker ECU can both hide its identity and shift the fingerprinting decision threshold gradually. Specifically, from the batch of n collected frames, only m of those are attack frames. The injection ratio $r = m/n$ can be carefully chosen for each step, so that the IDS cannot identify the attacker. More importantly, the fingerprinting decision threshold will be shifted via raising r iteratively, so that eventually, the attacker will be able to impersonate the legitimate ECU. As discussed in Sec. 7.3, the adversary can pursue the dominant impersonation or the complete impersonation. It will be demonstrated in the rest of this section that the former is easier to succeed, whereas the latter is stronger. This is summarized in an attack tree (Fig. 7.4).

In this section, we will use the Viden fingerprinting system [23] as our case study to demonstrate how multi-frame based schemes are vulnerable to the Hill-climbing-style attack. In the system, two fingerprinting schemes are employed independently in parallel: (i) Clock-based IDS (CIDS) [24] that tries to estimate the clock skews of different ECUs; and (ii) Voltage-based Identification (Viden) [23] that tries to abstract the voltage output

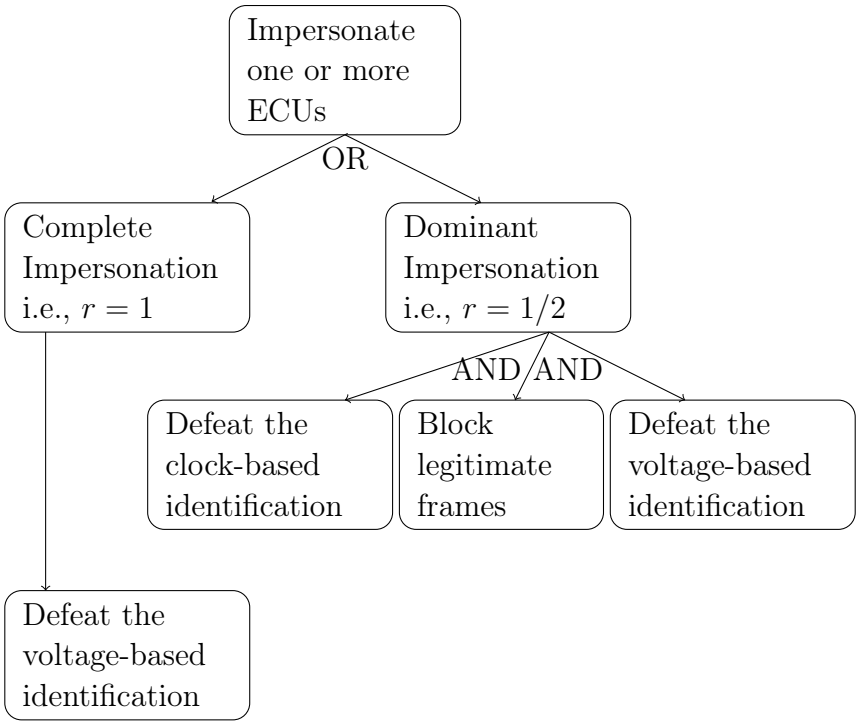


Figure 7.4: The attack tree against Viden's fingerprinting system.

characteristics. Both schemes collect a batch of frames, calculate the momentary features that ideally are constant over time, and then accumulate the latest feature with all historical ones. Because of the constancy of features, the cumulated quantity appears linear over time; hence the slope of it can be regarded as the profile of an ECU (i.e., the clock-skew S and the voltage profile Υ). Details of the two works are re-stated in Sec. 7.2.2. Here, we mainly focus on the fingerprinting system and we leave the discussion about the intrusion detection system to the end of this section.

7.4.1 Dominant Impersonation

In dominant impersonation (Fig. 7.5a), the attack frames are injected right after the legitimate frames so that the attack frames will be in charge of the vehicle for most of the time. With the carefully chosen injection ratio r for each step, the adversary will be able to evade identification. Note that CIDS does not work because the periodicity of frames is disrupted, thus the adversary only has to defeat the voltage-based scheme [23] at this point.

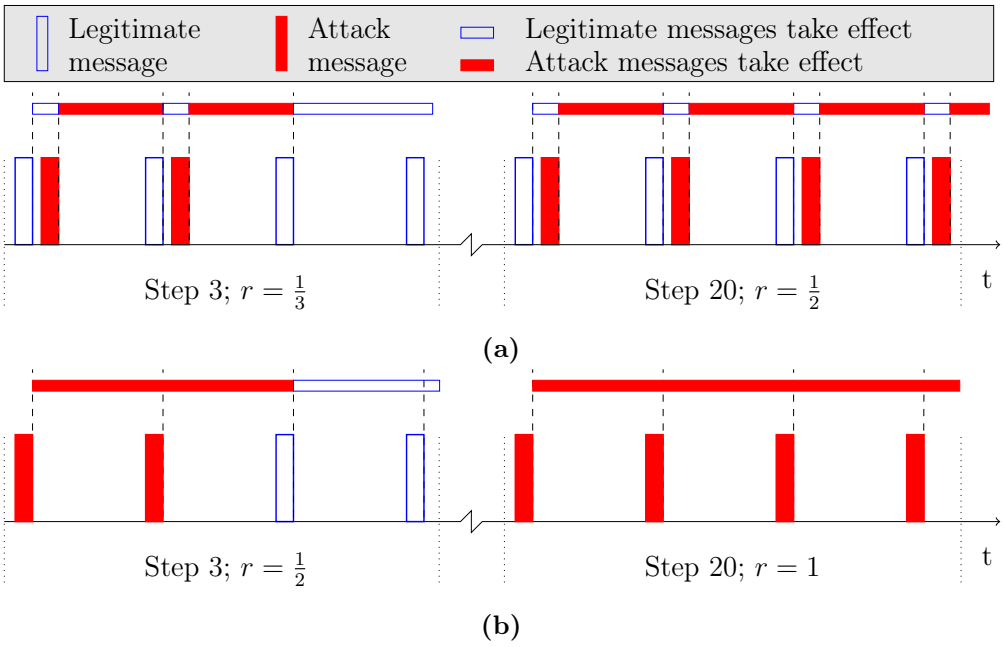


Figure 7.5: Hill-climbing-style attacks towards (a) Dominant impersonation and (b) Complete impersonation.

Defeating the voltage-based scheme Let us suppose the adversary compromises ECU $\mathbb{A}$ and runs the voltage-based fingerprinting scheme on it for k_0 steps. The adversary decides to start the intrusion at step $k_0 + 1$ to impersonate ECU $\mathbb{B}$. In each of these future steps, the voltage-based fingerprinting scheme will observe a batch of n *mixed frames*, consisting of m attack frames sent from ECU $\mathbb{A}$ and $n - m$ legitimate frames sent from ECU $\mathbb{B}$, both with $\mathbb{B}$'s target ID. The adversary wants to enlarge m as much as possible, while it does not want the fingerprinting scheme to identify $\mathbb{A}$. To do so, the injection ratio r has to be chosen carefully.

Given the learnt voltage profiles $\Upsilon_{\mathbb{A}}[k-1]$ and $\Upsilon_{\mathbb{B}}[k-1]$ from the previous $k-1$ steps, the adversary derives the desired voltage profile, $\Upsilon_{\mathbb{B}}[k]$, of the mixed frames for the next step (i.e., k-th step) by solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \left| \Upsilon_{\mathbb{B}}[k] - \frac{1}{2}\Big(\Upsilon_{\mathbb{A}}[k-1] + \Upsilon_{\mathbb{B}}[k-1]\Big) \right| \\ \text{subject to} \quad & \left| \Upsilon_{\mathbb{B}}[k] - \Upsilon_{\mathbb{B}}[k-1] \right| < \left| \Upsilon_{\mathbb{B}}[k] - \Upsilon_{\mathbb{A}}[k-1] \right|, \end{aligned} \tag{7.6}$$

where the objective is to make $\Upsilon_{\mathbb{B}}[k]$ as close to the threshold (the average of two profiles) as possible. With the constraint, the adversary makes sure that $\Upsilon_{\mathbb{B}}[k]$ is still closer to $\Upsilon_{\mathbb{B}}[k-1]$

than $\Upsilon_{\mathbb{A}}[k-1]$, so that the fingerprinting scheme regards $\mathbb{B}$ as the source of all the n frames. Eq. 7.6 yields

$$\Upsilon_{\mathbb{B}}[k] = \frac{1}{2}\Big(\Upsilon_{\mathbb{A}}[k-1] + \Upsilon_{\mathbb{B}}[k-1]\Big) + \epsilon \cdot \text{sign}(\Upsilon_{\mathbb{B}}[k-1] - \Upsilon_{\mathbb{A}}[k-1]), \tag{7.7}$$

where ϵ is a small enough factor for satisfying the constraint. See Fig. 7.6 for its illustration, from which we see that the profile of the mixed frames is sitting below the decision threshold by ϵ, fooling the fingerprinting scheme into believing that the intrusion source is $\mathbb{B}$. Furthermore, the threshold is shifted a bit closer to $\mathbb{A}$'s profile after each step, so is the profile of $\mathbb{B}$. This is done by gradually increasing the injection ratio r. The desired profile $\Upsilon_{\mathbb{B}}[k]$ depends on the injection ratio $r[k]$ of k-th step as

$$\Upsilon_{\mathbb{B}}[k] = r[k]\Upsilon_{\mathbb{A}}[k-1] + (1 - r[k])\,\Upsilon_{\mathbb{B}}[k_0], \tag{7.8}$$

where $\Upsilon_{\mathbb{B}}[k_0]$ is the ground truth profile of $\mathbb{B}$ because the intrusion started at step $k_0 + 1$; $\Upsilon_{\mathbb{B}}[k]$ is the profile of the n mixed frames at step k, and $k > k_0$. The detailed derivation of Finally we have the maximum injection ratio

$$r^*[k] = (\Upsilon_{\mathbb{B}}[k] - \Upsilon_{\mathbb{B}}[k_0])/(\Upsilon_{\mathbb{A}}[k-1] - \Upsilon_{\mathbb{B}}[k_0]). \tag{7.9}$$

With the maximum injection ratio, the adversary will be able to inject as many attack frames as possible at step k while avoiding identification. Since the legitimate ECU $\mathbb{B}$ is identified as the source, Viden accepts this batch of frames and updates $\Upsilon_{\mathbb{B}}$ to be a bit closer to $\Upsilon_{\mathbb{A}}$. With the updated profile, the adversary will be able to get an even higher r, thus injecting more and more attack frames at each future step. Hence, the injection ratio r will be able to reach 1/2. Note that the attack will not trigger the random forest classifier of Viden because the attack will not bring two profiles close to each other; instead, it is only the profile of the legitimate ECU that gets shifted (not the profile of the attacker ECU). The adversary can of course continue shifting the profile, i.e., r is approaching one. In the dominant impersonation, it is unnecessary to do so because it will not increase the amount of time that the attack frames take effect on the vehicle (Fig. 7.5a). However, it becomes

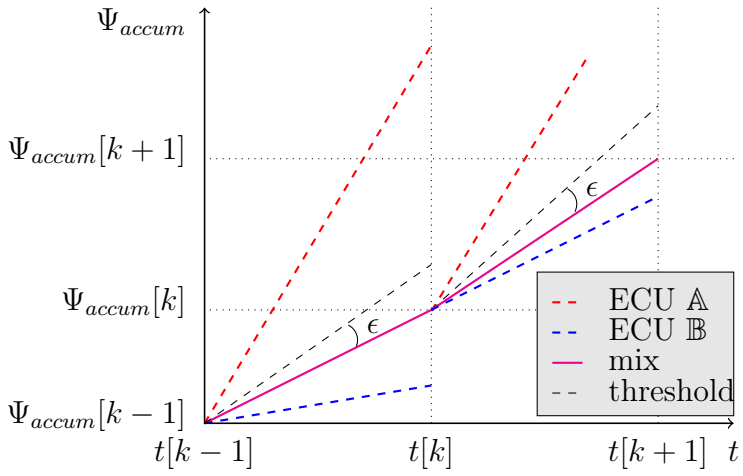


Figure 7.6: Hill-climbing-style attack.

necessary in the complete impersonation.$\Upsilon_{\mathbb{B}}$ will be eventually shifted to $\Upsilon_{\mathbb{A}}$. An illustration of such process is given in Figure 7.6. We use the same data samples as in Section 7.5 to simulate the attack and the results are plotted in Figure 7.12, where we can observe that r reaches $\frac{1}{2}$ at the fourth dot marker, while the attacker ECU evades Viden from identification all along.

7.4.2 Complete Impersonation

Although the dominant impersonation is strong enough to let the vehicle follow what the attack frames instruct, there are still some legitimate frames being transmitted on the CAN bus and instructing the vehicle to behave as normal. This conflict may cause suspicion. As a result, the adversary wants to block $\mathbb{B}$ from sending any frame, i.e., $r = 1$, by contending with the legitimate ECU $\mathbb{B}$ during the arbitration phase, using a forged and smaller identifier (than $\mathbb{B}$'s), making $\mathbb{B}$ lose the contention.

Blocking the legitimate frames In order to block the legitimate frame, the adversary has to know beforehand when the legitimate frame will be sent. To do so, the attacker ECU $\mathbb{A}$ learns the time skew $S_{\mathbb{AB}}$ of the legitimate ECU $\mathbb{B}$, just as a CIDS's ECU (say $\mathbb{C}$) that learns of $S_{\mathbb{CB}}$. After the learning phase where i legitimate frames have been sent, $\mathbb{A}$ calculates

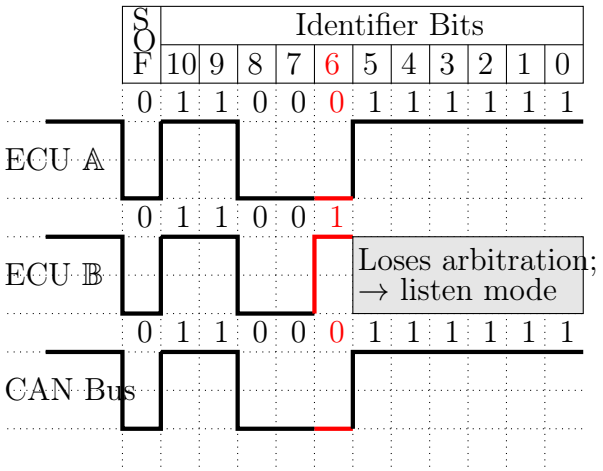


Figure 7.7: Contention

the moment when the next frame will be sent as

$$t_{\mathbb{A}}^{(i+1)} = t_{\mathbb{A}}^{(i)} + T \cdot (1 + S_{\mathbb{AB}}), \tag{7.10}$$

where T is the preset interval between two legitimate frames. Similar results were derived by two previous works [48]. The difference is that here the clock-skew imitation is used to block the legitimate frames, rather than directly inject the attack frames. An example of Eq. 7.10 is given in Fig. 7.8, in which $\mathbb{A}$ learns the time skew of $\mathbb{B}$ as $S_{\mathbb{AB}} = (10.03 - 10)/(10) = 0.003$. Meanwhile, CIDS learns it as $S_{\mathbb{CB}} = (10.01 - 10)/10 = 0.001$. According to Eq. 7.10, at the $\mathbb{A}$'s time $10.03 + 10 + 10 \cdot 0.003 = 20.06$ (or $\mathbb{B}$'s time $10 + 10 = 20$), $\mathbb{A}$ blocks the legitimate frame.

The blocking is done by taking advantage of the arbitration phase. In order to block a legitimate frame at $t_{\mathbb{A}}^{(i+1)}$, the attacker ECU $\mathbb{A}$ simply sends a frame with a smaller identifier than the legitimate ECU's. An illustration of the contention is shown. The contention will not trigger the "bit-error" because the CAN bus standard [114] does not count the wrong bits in the identifier field as bit-errors. Also, the feasibility of flipping a bit has been demonstrated by the *bus-off* attack where the adversary is able to disable a targeted ECU by flipping a bit in the data field of the frames sent by the targeted ECU [116].

Flooding the bus with high priority frames is a straightforward way of blocking, the countermeasures [254] against which will not work against our attack since Hoppe et al.

proposed several countermeasures using IDS against the naive flooding attack. Since our attack does not use flooding strategy, those countermeasures will not work in our case.

Defeating the clock-based scheme In addition to defeating the voltage-based fingerprinting scheme, now the adversary has to defeat the clock-based scheme [24] *at the same time* in the complete impersonation. This is because the periodicity of the frames still exists when the legitimate frames are blocked, and such periodicity can be used by CIDS to do identification. Since $t_{\mathbb{A}}^{(i+1)}$ has been occupied for blocking, the adversary has to inject the attack frame δ seconds later. The injection offset δ is illustrated in Fig. 7.8 and it can be determined as follows.

Similar to Sec. 7.4.1, the attacker ECU $\mathbb{A}$ has been learning the clock offset O as well as the clock skew S of the legitimate ECU $\mathbb{B}$ for $k-1$ steps. Given the maximum injection ratio $r^*[k]$ derived from Eq. 7.9, the maximum injection offset $\delta[k]$ can be obtained by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} \left| S_{\mathbb{B}}[k] - \frac{1}{2}\Big(S_{\mathbb{A}}[k-1] + S_{\mathbb{B}}[k-1]\Big)\right| \\ &\text{subject to} \left|S_{\mathbb{B}}[k] - S_{\mathbb{B}}[k-1]\right| < \left|S_{\mathbb{B}}[k] - S_{\mathbb{A}}[k-1]\right|. \end{aligned} \tag{7.11}$$

Similar to Eq. 7.6, the adversary tries to change $S_{\mathbb{B}}[k]$ as much as possible but not too far. In practice, because of the relativity of clock skew, $S_{\mathbb{A}}[k-1] = 0$ and $S_{\mathbb{B}}[k-1]$ is the clock skew from the perspective of $\mathbb{A}$. Eq. 7.11 yields the optimal desired clock skew $S_{\mathbb{B}}[k]$ that is

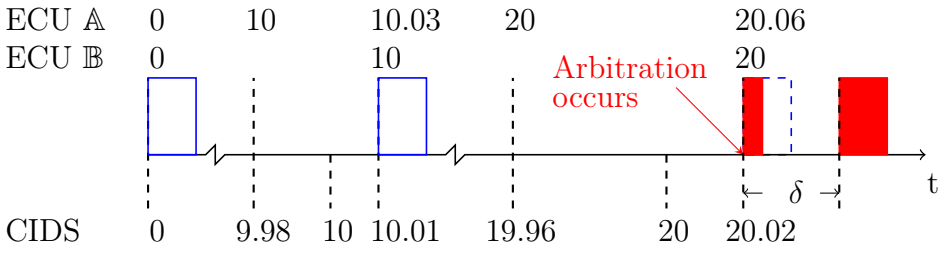


Figure 7.8: Complete impersonation zoomed-in from Fig. 7.5b. The three rows of numbers represent the time that different ECUs report.

similar to Eq. 7.7. Then, $\delta[k]$ can be derived from its relationship with $S_{\mathbb{B}}[k]$, which is

$$S_{\mathbb{B}}[k] = \{O[k-1] + \delta[k]r^*[k]n/(n-1)]\}/\{t[k] - t[k-1]\}, \tag{7.12}$$

where $O[k-1]$ is the clock offset of the previous step. Readers may go to to see why Eq. 7.12 holds.

It should be noticed that we choose such $\delta[k]$ in order to evade the identification of CIDS, not its detection. Of course, the adversary can always evade detection by adding the detection thresholds as constraints to Eq. 7.11. Here, however, we evaluate only the identification evasion due to experimental data limitations.

In Sec. 7.4, we showed that existing fingerprinting schemes are vulnerable to the Hill-climbing-style attack due to their dependence on multiple frames to make one detection or identification decision. In this section, we will describe our fingerprinting scheme, SIMPLE. Since it only requires a single CAN frame to perform the identification, it is immune to the Hill-climbing-style attack. SIMPLE operates in mainly five steps: (1) Feature extraction and FDA transformation, (2) Training feature templates for each ECU. (3) Associating a threshold to each ECU by comparing the feature templates using Mahalanobis distance metric (4) Identifying the origin of a single frame based on the proximity of its distance from the template of the device, to the device's predefined threshold. (5) Updating the feature templates of the devices after confirming validity of the origin of a message. Evaluation with real-world, in-vehicle data show that SIMPLE achieves 0.8985% equal error rate.

7.5 Attack Evaluation

We use real-world data collected from vehicles to evaluate the attacks against the voltage-based and the clock-based identification schemes. Results show that the attacker ECU was able to impersonate the legitimate ECU and was identified by neither of the schemes. Meanwhile, the profiles of the legitimate ECU was shifted to the attacker ECU profile gradually. Moreover, the number of steps required to achieve dominant impersonation increases with the difference between two profiles. Refer to Sec. 7.5.3 for the details of the attack evaluation results.

(a) Nissan Sentra 2016 **(b)** Subaru Outback 2011

Figure 7.9: Two experiment vehicles.

(a) CAN bus voltage measurement **(b)** CAN messages logging

Figure 7.10: Experimental setups.

7.5.1 In-vehicle Data Collection

We collected CAN messages from two vehicles, Nissan Sentra 2016 (Fig. 7.9a) and Subaru Outback 2011 (Fig. 7.9b), via the OBD-II port using a Tektronix DPO 3012 oscilloscope shown in Fig. 7.10a. The oscilloscope sampled at 50 Msps per channel with 8 bits of resolution and sent records to a computer via USB connection. We drove the vehicles for about forty minutes in each round, including local and highway, and collected over 16,000 frames. The in-vehicle CAN bus voltage dataset has been made public. In another round of data collection shown at Fig. 7.10b, we collected a million CAN messages with timestamps in microseconds using a PCAN-USB device and the python-can library while driving the vehicles for about ten minutes.

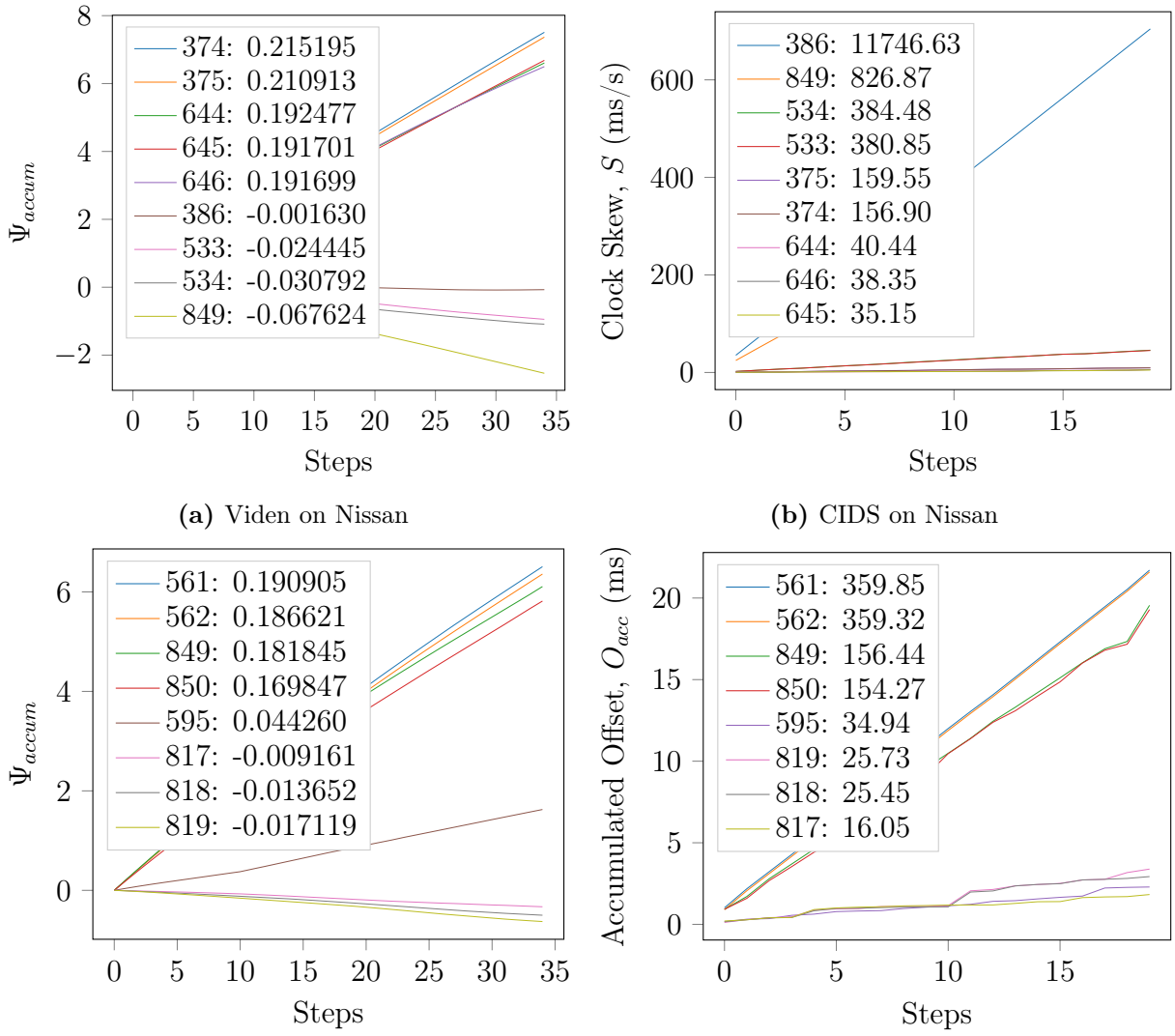
7.5.2 Ground truth establishment

In the data collected from real-world vehicles, frames were labelled with IDs only. Since one ECU can send messages with different IDs, we needed to associate these frames with ECUs as well (because SIMPLE aims to identify ECUs, not IDs). Hence we used Viden and CIDS to associate IDs with ECUs. Fig. 7.11 shows these results. For the Nissan Sentra, messages IDs fell into the ECU clusters of $\{374, 375\}$, $\{644, 645, 646\}$, $\{386\}$, $\{533, 534\}$, and $\{849\}$, which will be referred to as ECUs $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, $\mathbb{D}$, and $\mathbb{E}$ hereafter. For the Subaru Outback, IDs fell into $\{817, 818, 819, 820\}$, $\{593, 594, 595\}$, $\{849, 850, 855\}$, and $\{561, 562, 565\}$, which will be referred to as ECU $\mathbb{F}$ through $\mathbb{I}$. See Fig. 7.11 for the results.

7.5.3 Hill-climbing-style Attacks Results

We evaluated our attacks on both voltage-based and clock-based fingerprinting schemes, using the real-world data. We ran our attacks on ECUs $\mathbb{A}$ and $\mathbb{B}$'s data to demonstrate how $\mathbb{A}$ was able to impersonate $\mathbb{B}$. The results are plotted in Figs. 7.12 and 7.13. We can observe that $\mathbb{B}$'s profile was shifted gradually to $\mathbb{A}$'s while it never crossed the decision threshold. The dot markers indicate the moments when the number of attack frames, m was increased by one, i.e., injection ratio r was increased. In Fig. 7.13, the injection offset δ was also increased because $S_{\mathbb{B}}$ was increased along with r. We plot the results for 140 steps. But from the trend of r, we can see that it will approach one.

We also calculated the average of number of steps the attack needed to achieve dominant impersonation. We ran our attack on every pair of ECUs whose profiles were next to each other. Since there were multiple IDs associated with one ECU, we were able to calculate the average numbers of steps. Results show that the number of steps increased with the difference of voltage profiles between two ECUs. E.g, the difference of voltage profiles of ECUs $\mathbb{A}$ and $\mathbb{B}$ was 0.05; $\mathbb{A}$ impersonating $\mathbb{B}$ took averagely 115.5 steps. $\mathbb{B}$ impersonating $\mathbb{C}$ required 171.75 steps because their profiles' difference was 0.089. In summary, if the difference of the voltage profiles is large, the attack needs more steps to shift the profiles.



(a) Viden on Nissan

(b) CIDS on Nissan

(c) Viden on Subaru

(d) CIDS on Subaru

Figure 7.11: Identification results of the data from Nissan Sentra and Subaru Outback. Legends are sorted by profiles for clustering.

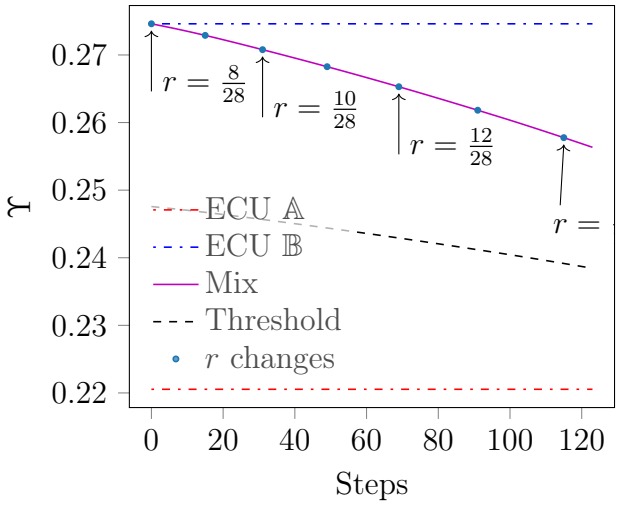


Figure 7.12: Dominant/complete impersonation against the voltage-based scheme

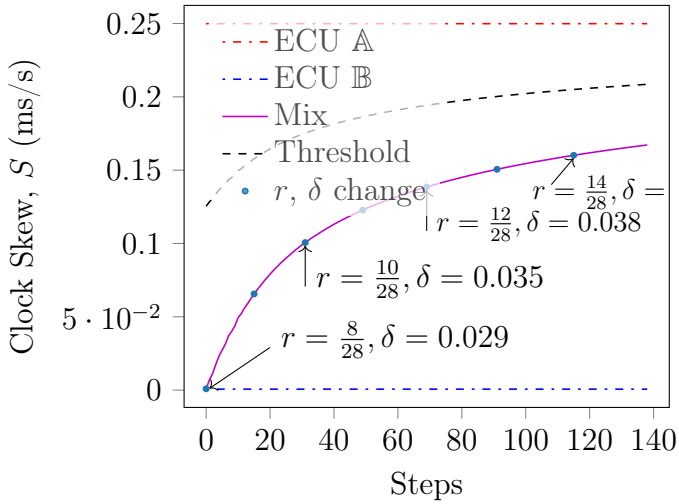


Figure 7.13: Complete impersonation against the clock-based scheme.

7.6 Discussion

Intrusion Detection System. In Viden's fingerprinting system [23] there is an underlying IDS that detects the intrusion and submits the suspicious frames to the fingerprinting schemes. However, Cho et al. did not specify what (type of) IDS is used, thus we do not specify the IDS in this work, either. Instead, we assume that as long as the fingerprinting result is the legitimate ECU, the vehicle will behave normally and no one gets suspicious. Note that if the IDS is based on multiple frames, it is also vulnerable to Hill-climbing-style attacks.

Arbitrary attack targets. In the previous example, we assumed that ECU 𝔸 and 𝔹's

profiles are adjacent. In general, there may be another ECU $\mathbb{C}$, whose profile is in between $\mathbb{A}$'s and $\mathbb{B}$'s, in which case Viden would regard $\mathbb{C}$ as the source of intrusion and an alarm may be triggered. To avoid so, the adversary can only impersonate the legitimate ECU that has the nearest voltage profile to the attacker ECU's profile. Such limitation can be addressed by enabling the adversary to imitate other ECU's voltage profile, e.g., changing the temperature. This can be achieved by extra code executions or increasing the CPU's clock speed. One way to achieve the imitation is to alter the CPU's clock speed; e.g., the adversary can increase/decrease the clock speed so that the temperature of the attack ECU will raise/drop, hence the voltage profile gets changed. Equipped with such capability, the adversary can first tune its voltage profile in a coarse-grained manner to become the nearest one to the target ECU's profile, and then mount our Hill-climbing-style attack (fine-grained) to impersonate the targeted ECU. We argue that even so, our adversary is not stronger than the *timing-voltage-aware* adversary [23]. We should also note that the Hill-climbing-style attack on the clock-based scheme does not suffer from such limitation because the imitation of clock-skew is not hard [23].

Retransmission. Because the CAN bus protocol [114] did not specify what an ECU would do if it loses a contention during the arbitration phase, we assumed in Sec. 7.4.2 that it would simply give up this frame. We argue that even if retransmission occurs once the CAN bus becomes idle in some specifications/implementations, the adversary can always choose to pursue the dominant impersonation, which does not involve any contention.

Voltage/timing profile knowledge. When the compromised ECU does not have an analog to digital converter (ADC), or a precise clock embedded, the adversary cannot calculate the optimal/maximum injection rate r. It can, however, be conservative. That is, it can find out the common minimum distance between profiles in car ECUs and use that as an empirical value to get a minimum injection rate. The attack will then take more time to succeed (or probabilistically). From our real-world experiment, however, r can still be high even with a small distance. See Fig. 7.12 for an example, where the distance is just 0.05 but r can still be as high as 8/28 at the first step.

7.7 Chapter Summary

We demonstrated the vulnerability of the existing multi-frame based automotive intrusion detection and identification systems to a Hill-climbing-style attack, which allows a compromised ECU to impersonate another. We also showed the feasibility of our attacks against Viden [24] and CIDS [23]. Our results show that the attack is practical and efficient because it took a short time to complete. Our attack motivated SIMPLE, a novel intrusion detection and identification system for in-vehicle networks that is immune to this type of attack. Our detection system uses physical layer features within a single frame to fingerprint the ECUs on a CAN bus, achieving an average EER close to 0% in in-lab, and 0.8985% based on the CAN bus data we collected from two real-world vehicles over an extensive period of time.

Chapter 8

Conclusion and Future Directions

In this dissertation, we studied the autonomous vehicle decision-making pipeline from the perspective of security. We proposed a hill-climbing-style attack that neutralizes SOTA CAN bus intrusion detection systems, and a perception attack called GhostImage that is physically-realizable thus being able to bypass the SOTA defenses against norm-bounded adversarial examples. To protect the autonomous system, we adopt the spatiotemporal consistency in two different, yet both effective ways: one is model-based, and another is data-driven. Both countermeasures are light-weight, and achieve outstanding trade-off between the true positive and false positive rates. The data-driven approach outperforms two SOTA defenses, while being extensible thanks to its general and simple design.

In Chapter 3, we presented GhostImage attacks against camera-based object classifiers. Using common optical effects, viz. lens flare/ghost effects, an attacker is able to inject arbitrary adversarial patterns into camera images using a projector. To increase the efficacy of the attack, we proposed a projector-camera channel model that predicts the location of ghosts, the resolution of the patterns in ghosts, given the projector-camera arrangement, and accounts for exposure control and color calibration. GhostImage attacks also leverage adversarial examples generation techniques to find optimal attack patterns. We evaluated GhostImage attacks using three image datasets and in both indoor and outdoor environments on three cameras, against both image classification and object detection. Experimental results show that GhostImage attacks achieve attack success rates as high as 100%, and also have potential impact on autonomous systems such as self-driving cars and surveillance systems.

In Chapter 4, we developed an attack detection algorithm that depends on whether there are ghost-object overlaps in images to detect the attacks, meanwhile eliminates false positives

via spatiotemporal consistency. We require only one frame to detect the existence of an attack and one more frame to make sure it is not a false positive. It detects failed attack attempt as well, making it suitable for attack prevention. Our detection algorithm yields worst-case EERs as high as 5% with real-world data. Our security analysis confirms that even an adaptive attacker cannot evade our detection due to the physical limitations.

In Chapter 5, we introduced the first approach to use spatiotemporal consistency of an object to detect misclassification attacks against the perception module of an autonomous system. We utilized a data-driven approach, which extracts spatiotemporal features from the output of the object tracker and cross-checks the consistency with the class label outputted by the object detector. To enhance the adversarial robustness, we incorporate additional context information, such as ego-vehicle velocity. We evaluate PercepGuard against both defense-unaware and defense-aware attacks, using a real-world and simulated datasets, and experiments. The FPR is as low as 5%, while the average TPR is as high as 99% from simulation and 96% from real-world experiments. PercepGuard is agnostic to attack methods and object detectors and is extensible. Our comparison with two SOTA defenses confirms the advantages of incorporating temporal features.

In Chpater 6, we proposed a framework for evaluating the impact of perception attacks on autonomous vehicle systems. In the framework, we intercepted the communication channel among the modules in the system such that we can hijack the messages on the fly. Using it, we discovered new vulnerabilities, include one that allows an attacker to compromise the safety by simply changing the object class. To remedy it, we designed a robust prediction system that can tolerates class uncertainty. Evaluation results show that our system is able to correct the prediction errors induced by misclassification attacks.

In Chapter 7, we demonstrated the vulnerability of the existing multi-frame based automotive intrusion detection and identification systems to a Hill-climbing-style attack, which allows a compromised ECU to impersonate another without being detected or identified, by carefully transmitting an increasing number of attack frames together with benign frames over time such that the fingerprint profile is shifted but remains belonging to the benign ECU. We demonstrated the feasibility of our attacks against Viden and CIDS, two SOTA CAN bus IDS's, using real-world CAN bus data collected from two vehicles by ourselves

over the course of two months. The attack motivated the design of a single-frame-based IDS, called SIMPLE, which is evaluated with our dataset. The dataset is published.

8.1 Future Directions

In the future, we could fingerprint an ECU by analyzing only one bit of the CAN frame. This will enable protection against the bit-flipping [262] or bus-off attack where the adversary flips one single bit of a number of frames so that the isolation mechanism of the CAN standard will be triggered (and as a result, the targeted ECU will be halted). We may also apply adversarial patch attacks with GhostImage attacks to achieve Trojan attacks. To develop a more robust perception attack detection algorithm, we may consider additional spatiotemporal features such as domain knowledge, detect other attacks such as creation attacks, and randomizing context where we create moving targets against attackers.

Following the consistency-based attack detection strategy, projector-based physical attacks, such as the phantom attack [22] and the attrack-zone attack [218], may be detected by texture analysis via active probing. Assuming a STOP is projected on a concrete wall by a phantom attacker. We may use mmWave radars to actively sense the material and texture from the STOP sign area [263]. A legitimate STOP sign should be made of metal with reflective paint on it. If the probing results do not match this, but rather show the texture of concrete, then it's likely a projected STOP sign. Similarly, visible light spectrum analysis can also be utilized, where a real STOP sign should reflect mostly red while the projected one does not.

Multiple cameras, providing various views of the scene, are able to enhance the perception robustness since cross-validation among them is viable in case some of them are compromised. Moreover, it is difficult for an adaptive attacker to compromise multiple cameras simultaneously. Such a multi-camera fusion method should largely enhance the adversarial robustness, especially when their configuration can be altered on the fly. The alteration can even be random to create a moving target for the attacker to further increase the attack difficulty.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] *NP Installation Series User's Manual*, NEC, 10 2007. [Online]. Available: https://www.projectorcentral.com/pdf/projector_manual_3978.pdf

[3] Canon, "Telephoto zoom ef-s 55-250mm."

[4] S. Hoory, T. Shapira, A. Shabtai, and Y. Elovici, "Dynamic adversarial patch for evading object detection models," *arXiv preprint arXiv:2010.13070*, 2020.

[5] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, T. Kohno, and D. Song, "Physical adversarial examples for object detectors," in *USENIX Conference on Offensive Technologies*, 2018.

[6] Y. Man, M. Li, and R. Gerdes, "Ghostimage: Remote perception attacks against camera-based image classification systems," in *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. USENIX Association, 2020.

[7] C. Zhou, Q. Yan, Y. Shi, and L. Sun, "Doublestar: Long-range attack towards depth estimation based obstacle avoidance in autonomous systems," in *USENIX Security Symposium*, 2022.

[8] 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," *arXiv preprint arXiv:2101.10011*, 2021.

[9] 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 Symposium on Security and Privacy*, 2021.

[10] 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 *European Conference on Computer Vision*. Springer, 2020.

[11] LG, "40lh5000," https://www.lg.com/us/tvs/lg-40LH5000-led-tv.

[12] Viofo, "A129 pro," https://www.viofo.com/en/home/149-viofo-a129-pro-duo-ultra-4k-front-full-hd-1080p-rear-dual-channel-wi-fi-gps-dash-camera.html.

[13] optoma, "Lv130," https://www.optoma.com/vn/product/lv130/.

[14] Tesla, "Autopilot," https://www.tesla.com/autopilot, 2020.

[15] Baidu, "Apollo," https://github.com/ApolloAuto/apollo.

[16] commaai, "openpilot," https://github.com/commaai/openpilot, 2021.

[17] Waymo, "Waymo," https://waymo.com, 2020.

[18] K. Xu, X. Xiao, J. Miao, and Q. Luo, "Data driven prediction architecture for autonomous driving and its application on apollo platform," in *2020 IEEE Intelligent Vehicles Symposium (IV)*.

[19] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 39–57.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[21] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[22] 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 SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[23] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1109–1123.

[24] ——, "Fingerprinting electronic control units for vehicle intrusion detection." in *USENIX Security Symposium*, 2016, pp. 911–927.

[25] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[26] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, 2016.

[27] K. N. Truong, S. N. Patel, J. W. Summet, and G. D. Abowd, "Preventing camera recording by designing a capture-resistant environment," in *International conference on ubiquitous computing*. Springer, 2005, pp. 73–86.

[28] E. Ribnick, S. Atev, O. Masoud, N. Papanikolopoulos, and R. Voyles, "Real-time detection of camera tampering," in *2006 IEEE International Conference on Video and Signal Based Surveillance*. IEEE, 2006, pp. 10–10.

[29] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "Sok: A minimalist approach to formalizing analog sensor security," in *2020 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2020, pp. 480–495. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2020.00026

[30] I. Giechaskiel and K. B. Rasmussen, "Taxonomy and challenges of out-of-band signal injection attacks and defenses," *IEEE Communication Surveys & Tutorials*, 2020.

[31] 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 Symposium*, 2020.

[32] 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 Symposium*, 2020.

[33] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "Savior: Securing autonomous vehicles with robust physical invariants," in *USENIX Security Symposium*, 2020.

[34] C. Xiao, R. Deng, B. Li, T. Lee, B. Edwards, J. Yi, D. Song, M. Liu, and I. Molloy, "Advit: Adversarial frames identifier based on temporal consistency in videos," in *IEEE International Conference on Computer Vision*, 2019.

[35] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking," in *USENIX Security Symposium*, 2021.

[36] J. Liu and J. Park, ""seeing is not always believing": Detecting perception error attacks against autonomous vehicles," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[37] J. Zhang, Y. Zhang, K. Lu, J. Wang, K. Wu, X. Jia, and B. Liu, "Detecting and identifying optical signal attacks on autonomous driving systems," *IEEE Internet of Things Journal*, 2021.

[38] N. M. Gürel, X. Qi, L. Rimanic, C. Zhang, and B. Li, "Knowledge enhanced machine learning pipeline against diverse adversarial attacks," in *International Conference on Machine Learning*, 2021.

[39] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2016, pp. 582–597.

[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[41] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[42] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 447–462.

[43] 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 Symposium*. San Francisco, 2011.

[44] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 787–800.

[45] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

[46] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "Nic: Detecting adversarial samples with neural network invariant checking." in *Network and Distributed System Security Symposium*, 2019.

[47] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "Simple: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 229–244.

[48] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, 2018.

[49] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[50] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[51] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[52] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[53] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[54] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *Proceedings of the IEEE conference on learning representations*, 2018.

[55] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *European Conference on Computer Vision*. Springer, 2018, pp. 158–174.

[56] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[57] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 506–519.

[58] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proceedings of the International Conference on Learning Representations*, 2016.

[59] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *Proceedings of the IEEE conference on learning representations*, 2018.

[60] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1528–1540.

[61] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, "Invisible mask: Practical attacks on face recognition with infrared," *arXiv preprint arXiv:1803.04683*, 2018.

[62] C. Sitawarin, A. N. Bhagoji, A. Mosenia, P. Mittal, and M. Chiang, "Rogue signs: Deceiving traffic sign recognition with malicious ads and logos," in *IEEE Security and Privacy Workshop on Deep Learning and Security*. IEEE, 2018.

[63] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2019, pp. 1989–2004.

[64] Z. Wu, S.-N. Lim, L. Davis, and T. Goldstein, "Making an invisibility cloak: Real world adversarial attacks on object detectors," *arXiv preprint arXiv:1910.14667*, 2019.

[65] S. Komkov and A. Petiushko, "Advhat: Real-world adversarial attack on arcface face id system," *arXiv preprint arXiv:1908.08705*, 2019.

[66] G. S. Dhillon, K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1uR4GZRZ

[67] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk9yuql0Z

[68] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.

[69] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, 2018.

[70] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Bys4ob-Rb

[71] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 278–287.

[72] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *8th International Conference on Learning Representations (ICLR)*, 2020.

[73] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed Systems Security Symposium (NDSS)*, 2018.

[74] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," 2017.

[75] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proceedings of the IEEE conference on learning representations*, 2017.

[76] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[77] F. Cai, J. Li, and X. Koutsoukos, "Detecting adversarial examples in learning-enabled cyber-physical systems using variational autoencoder for regression," in *Workshop on Assured Autonomous Systems*, 2020.

[78] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S18Su--CW

[79] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyJ7ClWCb

[80] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2019, pp. 2267–2281.

[81] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Jul. 2018. [Online]. Available: https://arxiv.org/abs/1802.00420

[82] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. 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 Symposium on Security and Privacy*, 2021.

[83] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "SLAP: Improving physical adversarial examples with short-lived adversarial perturbations," in *USENIX Security Symposium*, 2021.

[84] 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 Workshop on IoT*. IEEE, 2019.

[85] J. B. Li, F. R. Schmidt, and J. Z. Kolter, "Adversarial camera stickers: A physical camera attack on deep learning classifier," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2019.

[86] B. Nassi, D. Nassi, R. Ben-Netanel, Y. Mirsky, O. Drokin, and Y. Elovici, "Phantom of the adas: Phantom attacks on driver-assistance systems."

[87] L. Nguyen, S. S. Arora, Y. Wu, and H. Yang, "Adversarial light projection attacks on face recognition systems: A feasibility study," 2020.

[88] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 445–467.

[89] 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," 2020.

[90] 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*, 2020.

[91] J.-P. Schulze, P. Sperl, and K. Böttinger, "Da3g: Detecting adversarial attacks by analysing gradients," in *European Symposium on Research in Computer Security*. Springer, 2021.

[92] A. P. Anand, H. Gokul, H. Srinivasan, P. Vijay, and V. Vijayaraghavan, "Adversarial patch defense for optical flow networks in video action recognition," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020.

[93] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *International Conference on Learning Representations (ICLR)*, 2020.

[94] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *31st Conference on Neural Information Processing Systems*, 2017.

[95] M. Yin, S. Li, Z. Cai, C. Song, M. S. Asif, A. K. Roy-Chowdhury, and S. V. Krishnamurthy, "Exploiting multi-object relationships for detecting adversarial attacks in complex scenes," in *International Conference on Computer Vision*, 2021.

[96] M. Yin, S. Li, C. Song, M. S. Asif, A. K. Roy-Chowdhury, and S. V. Krishnamurthy, "Adc: Adversarial attacks against object detection that evade context consistency checks," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[97] M. Sun, Z. Li, C. Xiao, H. Qiu, B. Kailkhura, M. Liu, and B. Li, "Can shape structure features improve model robustness under diverse adversarial settings?" in *IEEE/CVF International Conference on Computer Vision*, 2021.

[98] X. Wang, S. Li, M. Liu, Y. Wang, and A. K. Roy-Chowdhury, "Multi-expert adversarial attack detection in person re-identification using context inconsistency," in *International Conference on Computer Vision*, 2021.

[99] A. Amich and B. Eshete, "Morphence: Moving target defense against adversarial examples," in *Annual Computer Security Applications Conference*, 2021.

[100] S. Abdelnabi and M. Fritz, ""what's in the box?!": Deflecting adversarial attacks by randomly deploying adversarially-disjoint models," *arXiv preprint arXiv:2102.05104*, 2021.

[101] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, "Learning human identity from motion patterns," *IEEE Access*, vol. 4, 2016.

[102] S. Vhaduri and C. Poellabauer, "Multi-modal biometric-based implicit authentication of wearable device users," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, 2019.

[103] C. Wu, K. He, J. Chen, Z. Zhao, and R. Du, "Liveness is not enough: Enhancing fingerprint authentication with behavioral biometrics to defeat puppet attacks," in *USENIX Security Symposium*, 2020.

[104] S. Mekruksavanich and A. Jitpattanakul, "Deep learning approaches for continuous authentication based on activity patterns using mobile sensing," *Sensors*, vol. 21, no. 22, 2021.

[105] A. Acar, H. Aksu, A. S. Uluagac, and K. Akkaya, "A usable and robust continuous authentication framework using wearables," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2140–2153, 2020.

[106] M. Zhou, Q. Wang, X. Lin, Y. Zhao, P. Jiang, Q. Li, C. Shen, and C. Wang, "Presspin: Enabling secure pin authentication on mobile devices via structure-borne sounds," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[107] N. K. Shaydyuk and T. Cleland, "Biometric identification via retina scanning with liveness detection using speckle contrast imaging," in *IEEE International Carnahan Conference on Security Technology*, 2016.

[108] S. K. Yadav, K. Tiwari, H. M. Pandey, and S. A. Akbar, "A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions," *Knowledge-Based Systems*, vol. 223, 2021.

[109] L. Mou, C. Zhou, P. Zhao, B. Nakisa, M. N. Rastgoo, R. Jain, and W. Gao, "Driver stress detection via multimodal fusion using attention-based cnn-lstm," *Expert Systems with Applications*, vol. 173, 2021.

[110] H. F. Nweke, Y. W. Teh, G. Mujtaba, and M. A. Al-Garadi, "Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions," *Information Fusion*, vol. 46, 2019.

[111] J. Zhang, Z. Yin, P. Chen, and S. Nichele, "Emotion recognition using multi-modal data and machine learning techniques: A tutorial and review," *Information Fusion*, vol. 59, 2020.

[112] X. Zhou, W. Liang, I. Kevin, K. Wang, H. Wang, L. T. Yang, and Q. Jin, "Deep-learning-enhanced human activity recognition for internet of healthcare things," *IEEE Internet of Things Journal*, vol. 7, no. 7, 2020.

[113] H. Li, A. Shrestha, H. Heidari, J. Le Kernec, and F. Fioranelli, "Bi-lstm network for multimodal continuous human activity recognition and fall detection," *IEEE Sensors Journal*, vol. 20, no. 3, 2019.

[114] R. Bosch, "CAN specification v2.0," Bosch, Tech. Rep., 1991.

[115] A. Greenberg, "Hackers remotely kill a jeep on the highway—with me in it," *Wired*, Dec 2015.

[116] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1044–1055.

[117] R. M. Gerdes, M. Mina, S. F. Russell, and T. E. Daniels, "Physical-layer identification of wired ethernet devices," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1339–1353, Aug 2012.

[118] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 1110–1115.

[119] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. ACM, 2017, p. 11.

[120] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. IEEE, 2010, pp. 92–98.

[121] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 130–139.

[122] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown can bus networks," *Vehicular Communications*, vol. 9, pp. 43–52, 2017.

[123] X. Ying, G. Bernieri, M. Conti, and R. Poovendran, "Tacan: Transmitter authentication through covert channels in controller area networks," *arXiv preprint arXiv:1903.05231*, 2019.

[124] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," *arXiv preprint arXiv:1710.02692*, 2017.

[125] O. Avatefipour, A. Hafeez, M. Tayyab, and H. Malik, "Linking received packet to the transmitter through physical-fingerprinting of controller area network," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2017, pp. 1–6.

[126] T. Dagan and A. Wool, "Parrot, a software-only anti-spoofing defense system for the can bus," in *14th Embedded Security in Cars*, 2016.

[127] P.-S. Murvay and B. Groza, "Source identification using signal characteristics in controller area networks," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 395–399, 2014.

[128] Amazon, "Prime air delivery."

[129] Google, "Nest and google home. now under one roof," nest.com, 2020.

[130] Amazon, "Ring," ring.com, 2020.

[131] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 95–110.

[132] J. Selvaraj, G. Y. Dayanıklı, N. P. Gaunkar, D. Ware, R. M. Gerdes, M. Mina *et al.*, "Electromagnetic induction attacks against embedded systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 499–510.

[133] T. Sugawara, B. Cyr, S. Rampazzi, D. Genkin, and K. Fu, "Light commands: Laser-based audio injection attacks on voice-controllable systems."

[134] 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 *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 881–896.

[135] Q. Yan, K. Liu, Q. Zhou, H. Guo, and N. Zhang, "Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided wave," in *Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[136] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 540–555, 2013.

[137] M. B. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, "Physically-based real-time lens flare rendering," *ACM Trans. Graph. (Proc. SIGGRAPH 2011)*, vol. 30, no. 4, pp. 108:1–108:9, 2011.

[138] K. MIZOKAMI, "China could blind u.s. satellites with lasers," https://www.popularmechanics.com/military/weapons/a29307535/china-satellite-laser-blinding/, 2019.

[139] P. Vitoria and C. Ballester, "Automatic flare spot artifact detection and removal in photographs," *Journal of Mathematical Imaging and Vision*, vol. 61, no. 4, pp. 515–533, 2019.

[140] H.-C. Lee, *Introduction to color imaging science*. Cambridge University Press, 2005.

[141] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72.

[142] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC Conference*

on Computer and Communications Security, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 103–117. [Online]. Available: https://doi.org/10.1145/3133956.3134052

[143] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[144]

[145] *MT9M034 1/3-Inch CMOS Digital Image Sensor*, ON semiconductor.

[146] NEC, "Np05zl, 4.62–7.02:1 zoom lens."

[147] Opteka, "Opteka 650-1300mm telephoto zoom lens."

[148] Ring, "Indoor security cameras," https://shop.ring.com/collections/security-cams#indoor, 2019.

[149] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[150] Y. Man, M. Li, and R. Gerdes, "Poster: Perceived adversarial examples," in *IEEE Symposium on Security and Privacy*, 2019.

[151] *NP Installation Series Specification Sheet*, NEC, 11 2009. [Online]. Available: https://www.projectorcentral.com/pdf/projector_spec_3978.pdf

[152] A. Saha, A. Subramanya, K. Patil, and H. Pirsiavash, "Role of spatial context in adversarial robustness for object detection," in *CVPR 2020 Workshop on Adversarial Machine Learning in Computer Vision*, 2020.

[153] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[154] *NVIDIA TESLA P100 GPU ACCELERATOR*, Nvidia, 2016.

[155] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[156] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[157] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[158] Barco, "Xdl-4k75."

[159] *MT9V034 1/3-Inch Wide-VGA CMOS Digital Image Sensor*, ON semiconductor, 2017. [Online]. Available: https://www.onsemi.com/pub/Collateral/MT9V034-D.PDF

[160] Ring, "Standard and advanced motion detection systems used in ring devices," https://support.ring.com/hc/en-us/articles/115005914666-Standard-and-Advanced-Motion-Detection-Systems-Used-in-Ring-Devices, 2020.

[161] YunYang1994, "tensorflow-yolov3," https://github.com/YunYang1994/tensorflow-yolov3, 2020.

[162] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014.

[163] Epson, "Pro l1490u wuxga 3lcd laser projector."

[164] Apple, "iphone 8 plus horrible lens flare and reflections," https://discussions.apple.com/thread/8118139, 2017.

[165] ——, "iphone 11 pro's irritating lens flare problem," https://www.reddit.com/r/apple/comments/da6qft/iphone_11_pros_irritating_lens_flare_problem/, 2019.

[166] MasterClass, "What is lens flare photography? tips and tricks for achieving perfect lens flare," https://www.masterclass.com/articles/what-is-lens-flare-photography-tips-

and-tricks-for-achieving-perfect-lens-flare#whats-the-difference-between-lens-flare-and-bokeh, 2019.

[167] C. Burchby, "A brief history of the lens flares technique — and the end of film," https://www.laweekly.com/a-brief-history-of-the-lens-flares-technique-and-the-end-of-film/, 2013.

[168] Wikipedia, "Bloom (shader effect)," https://en.wikipedia.org/wiki/Bloom_(shader_effect), 2020.

[169] Y. Chen, X. Yuan, J. Zhang, Y. Zhao, S. Zhang, K. Chen, and X. Wang, "Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[170] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno, "Physical adversarial examples for object detectors," in *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[171] L. World, "Rgb laser," https://www.laserworld.com/en/show-laser-light-faq/glossary-definitions/88-r/2549-rgb-laser.html.

[172] C. Carlberg, "Hexbright, an open source light," https://www.kickstarter.com/projects/christian-carlberg/hexbright-an-open-source-light.

[173] Y. Man, M. Li, and R. Gerdes, "Ghostimage: Perception domain attacks against vision-based object classification systems," *arXiv preprint arXiv:2001.07792*, 2020.

[174] Y. Xu, H. Nagahara, A. Shimada, and R.-i. Taniguchi, "Transcut: Transparent object segmentation from a light-field image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3442–3450.

[175] E. Optics, "Introduction to liquid lenses," https://www.edmundoptics.com/knowledge-center/application-notes/imaging/introduction-to-liquid-lenses/.

[176] J. Zhang, Y. Zhang, K. Lu, J. Wang, K. Wu, X. Jia, and B. Liu, "Detecting and identifying optical signal attacks on autonomous driving systems," *IEEE Internet of Things Journal*, 2020.

[177] F. Chabert, "Automated lens flare removal," Technical report, Stanford University, Department of Electrical Engineering, Tech. Rep., 2015.

[178] Y. Wu, Q. He, T. Xue, R. Garg, J. Chen, A. Veeraraghavan, and J. Barron, "Single-image lens flare removal," *arXiv preprint arXiv:2011.12485*, 2020.

[179] A. Rosebrock, "Detecting multiple bright spots in an image with python and opencv," https://www.pyimagesearch.com/2016/10/31/detecting-multiple-bright-spots-in-an-image-with-python-and-opencv/, 2016.

[180] eufy, "Indoor cam 2k pan and tilt," 2021.

[181] Wikipedia, "Pan–tilt–zoom camera," https://en.wikipedia.org/wiki/Pan-tilt-zoom-camera, 2021.

[182] A. R. DiDonato and M. P. Jarnagin, "Integration of the general bivariate gaussian distribution over an offset circle," *Mathematics of Computation*, vol. 15, pp. 375–382, 1961.

[183] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[184] S. Lombardi and K. Nishino, "Reflectance and illumination recovery in the wild," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 129–141, 2015.

[185] Starship, "The self-driving delivery robot," https:/www.starship.xyz.

[186] L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu, "Universal physical camouflage attacks on object detectors," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[187] S. Köhler, R. Baker, and I. Martinovic, "Signal injection attacks against ccd image sensors," *arXiv preprint arXiv:2108.08881*, 2021.

[188] Carla, "Carla simulator," https://github.com/carla-simulator/carla.

[189] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[190] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*, 2016.

[191] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *IEEE international conference on computer vision*, 2015.

[192] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.

[193] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016.

[194] Tesla, "Autopilot," https://www.tesla.com/autopilot, 2020.

[195] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems: Theory and practice," *ACM Transactions on Computer Systems (TOCS)*, vol. 10, no. 4, 1992.

[196] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *IEEE International Conference on Computer Vision*, 2007.

[197] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[198] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[199] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*. Springer, 2012.

[200] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and M. Lienkamp, "A deep learning-based radar and camera sensor fusion architecture for object detection," in *IEEE Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2019.

[201] B. Shahian Jahromi, T. Tulabandhula, and S. Cetin, "Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles," *Sensors*, vol. 19, no. 20, 2019.

[202] Apple, "Maps," https://www.apple.com/maps/.

[203] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[204] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *USENIX symposium on operating systems design and implementation (OSDI)*, 2016.

[205] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes, "Drivetruth: Automated autonomous driving dataset generation for security applications," in *The Automotive and Autonomous Vehicle Secuirty Workshop*, 2022.

[206] S. Casper, M. Nadeau, and G. Kreiman, "One thing to fool them all: Generating interpretable, universal, and physically-realizable adversarial features," in *International Conference on Learning Representations*, 2022.

[207] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14.

[208] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[209] Y. Ma, J. Sharp, R. Wang, E. Fernandes, and X. Zhu, "Sequential attacks on kalman filter-based forward collision warning systems," in *AAAI Conference on Artificial Intelligence*, 2021.

[210] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[211] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE conference on computer vision and pattern recognition*, 2017.

[212] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR*, 2016.

[213] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, 2010.

[214] Mount-It!, "Tailgate tv hitch mount," https://mount-it.com/products/mount-it-tailgate-tv-trailer-hitch-mount-mi-380.

[215] LG, "75" xs4g-b series uhd window facing high brightness display," https://www.lg.com/us/business/outdoor-displays/lg-75xs4g-b.

[216] Epson, "Pro l1490u," https://www.focusedtechnology.com/epson-v11ha16020-projector.html.

[217] M. Sun, Z. Li, C. Xiao, H. Qiu, B. Kailkhura, M. Liu, and B. Li, "Can shape structure features improve model robustness under diverse adversarial settings?" in *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

[218] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes, "Physical hijacking attacks against object trackers," in *ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[219] Y. Man, R. Muller, M. Li, Z. B. Celik, and R. Gerdes, "That person behaves like a car: Misclassification attack detection for autonomous systems using spatiotemporal consistency," in *USENIX Security Symposium*, 2023.

[220] S. Khaitan, Q. Lin, and J. M. Dolan, "Safe planning and control under uncertainty for self-driving," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 9826–9837, 2021.

[221] J. Chandiramani, "Decision making under uncertainty for automated vehicles in urban situations," 2017.

[222] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *IEEE International Conference on Robotics and Automation*, 2011.

[223] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On adversarial robustness of trajectory prediction for autonomous vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[224] Y. Man, R. Muller, M. Li, Z. B. Celik, and R. Gerdes, "Evaluating perception attacks on prediction and planning of autonomous vehicles," in *USENIX Security Symposium Poster Session*, 2022.

[225] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*. Springer, 2020, pp. 683–700.

[226] M. Naghshvar, A. K. Sadek, and A. J. Wiggers, "Risk-averse behavior planning for autonomous driving under uncertainty," in *Machine Learning for Intelligent Transportation Systems Workshop*, 2018.

[227] W. Xu, J. Pan, J. Wei, and J. M. Dolan, "Motion planning under uncertainty for on-road autonomous driving," in *IEEE International Conference on Robotics and Automation*, 2014.

[228] S. Kabir, I. Sorokos, K. Aslansefat, Y. Papadopoulos, Y. Gheraibia, J. Reich, M. Saimler, and R. Wei, "A runtime safety analysis concept for open adaptive systems," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2019, pp. 332–346.

[229] V. Gautam, Y. Gheraibia, R. Alexander, and R. D. Hawkins, "Runtime decision making under uncertainty in autonomous vehicles," in *Proceedings of the Workshop on Artificial Intelligence Safety*, 2021.

[230] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty—a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[231] K. Aslansefat, I. Sorokos, D. Whiting, R. Tavakoli Kolagari, and Y. Papadopoulos, "Safeml: safety monitoring of machine learning classifiers through statistical difference measures," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2020, pp. 197–211.

[232] K. Aslansefat, S. Kabir, A. Abdullatif, V. Vasudevan, and Y. Papadopoulos, "Toward improving confidence in autonomous vehicle software: A study on traffic sign recognition systems," *Computer*, vol. 54, no. 8, pp. 66–76, 2021.

[233] T. A. Foundation, "Autoware," https://github.com/autowarefoundation/autoware.

[234] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[235] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," *arXiv preprint arXiv:1907.07792*, 2019.

[236] R. Jiao, X. Liu, T. Sato, Q. A. Chen, and Q. Zhu, "Semi-supervised semantics-guided adversarial training for trajectory prediction," *arXiv preprint arXiv:2205.14230*, 2022.

[237] Z. Hu, J. Shen, S. Guo, X. Zhang, Z. Zhong, Q. A. Chen, and K. Li, "Pass: A system-driven evaluation platform for autonomous driving safety and security," in *The Automotive and Autonomous Vehicle Secuirty Workshop*, 2022.

[238] Z. Wan, J. Shen, J. Chuang, X. Xia, J. Garcia, J. Ma, and Q. A. Chen, "Too afraid to drive: Systematic discovery of semantic dos vulnerability in autonomous driving planning under physical-world attacks," in *Network and Distributed System Security Symposium*, 2022.

[239] Y. Man, J. Deng, G. T. Amariucai, and S. Wei, "Hquad: Statistics of hamiltonian cycles in wireless rechargeable sensor networks," in *2018 25th International Conference on Telecommunications (ICT)*. IEEE, 2018, pp. 549–553.

[240] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[241] V. Adeli, M. Ehsanpour, I. Reid, J. C. Niebles, S. Savarese, E. Adeli, and H. Rezatofighi, "Tripod: Human trajectory and pose dynamics forecasting in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[242] J. Zhong, H. Sun, W. Cao, and Z. He, "Pedestrian motion trajectory prediction with stereo-based 3d deep pose estimation and trajectory learning," *IEEE access*, vol. 8, pp. 23 480–23 486, 2020.

[243] B. Barrois and C. Wöhler, "3d pose estimation of vehicles using stereo camera," in *Transportation Technologies for Sustainability*. Springer, 2013, pp. 1–24.

[244] 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*, 2020, pp. 65–76.

[245] M. Sun, M. Li, and R. Gerdes, "Truth-aware optimal decision-making framework with driver preferences for v2v communications," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–9.

[246] L. Yao, Y. Man, Z. Huang, J. Deng, and X. Wang, "Secure routing based on social similarity in opportunistic networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 594–605, 2015.

[247] M. Sun, M. Li, and R. Gerdes, "A data trust framework for vanets enabling false data detection and secure vehicle tracking," in *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2017, pp. 1–9.

[248] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," in *Workshop on Embedded Security in Cars*, 2004.

[249] S. Fröschle and A. Stühring, "Analyzing the capabilities of the can attacker," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 464–482.

[250] A. Costley, C. Kunz, R. Gerdes, and R. Sharma, "Low cost, open-source testbed to enable full-sized automated vehicle research," *arXiv preprint arXiv:1708.07771*, 2017.

[251] T. Ziermann, S. Wildermann, and J. Teich, "Can+: A new backward-compatible controller area network (can) protocol with up to 16× higher data rate," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* IEEE, 2009, pp. 1088–1093.

[252] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th.* IEEE, 2008, pp. 1–5.

[253] C. J. Szilagyi, "Low cost multicast network authentication for embedded control systems," Ph.D. dissertation, Carnegie Mellon University, 2012.

[254] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks—practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11–25, 2011.

[255] B. Groza and P.-S. Murvay, "Security solutions for the controller area network: Bringing authentication to in-vehicle networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, 2018.

[256] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[257] B. Danev, D. Zanetti, and S. Capkun, "On physical-layer identification of wireless devices," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 6, 2012.

[258] R. M. Gerdes and S. Mallick, "Physical-layer detection of hardware keyloggers," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 26–47.

[259] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006. [Online]. Available: http://research.microsoft.com/en-us/um/people/cmbishop/prml/

[260] C. Soutar *et al.*, "Biometric system security," *White Paper, Bioscrypt, http://www. bioscrypt. com*, 2002.

[261] L. Technologies, "Ltc1743—12-bit, 50msps adc," 2018. [Online]. Available: https://www.analog.com/en/products/ltc1743.html#product-documentation

[262] A. Z. Mohammed, Y. Man, R. Gerdes, M. Li, and Z. B. Celik, "Physical layer data manipulation attacks on the can bus," in *Automotive and Autonomous Vehicle Security (AutoSec) Workshop*, 2022.

[263] P. Kumari, J. Choi, N. González-Prelcic, and R. W. Heath, "Ieee 802.11 ad-based radar: An approach to joint vehicular communication-radar system," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3012–3027, 2017.