VisionGuard: Secure and Robust Visual Perception of Autonomous Vehicles in Practice

Xingshuo Han
xingshuo001@e.ntu.edu.sg
Nanyang Technological University
Singapore

Haozhao Wang*
hz_wang@hust.edu.cn
Huazhong University of Science and Technology
China

Kangqiao Zhao
KANGQIAO002@e.ntu.edu.sg
Nanyang Technological University
Singapore

Gelei Deng
GDENG003@e.ntu.edu.sg
Nanyang Technological University
Singapore

Yuan Xu
xu.yuan@ntu.edu.sg
Nanyang Technological University
Singapore

Hangcheng Liu
hangcheng.liu@ntu.edu.sg
Nanyang Technological University
Singapore

Han Qiu
qiuhan@tsinghua.edu.cn
Tsinghua University
China

Tianwei Zhang
tianwei.zhang@ntu.edu.sg
Nanyang Technological University
Singapore

Abstract

Modern Autonomous Vehicles (AVs) implement the Visual Perception Module (VPM) to perceive their surroundings. This VPM adopts various Deep Neural Network (DNN) models to process the data collected from cameras and LiDAR. Prior studies have shown that these models are vulnerable to physical adversarial examples (PAEs), which pose a critical safety risk to the autonomous driving task. While a few defense methods have been proposed to safeguard AVs, most of them only target a limited set of attack types and specific scenarios, making them impractical for real-world protection.

In this paper, we introduce VisionGuard, a novel and practical methodology to comprehensively detect and mitigate various types of PAEs to the VPM. The key of VisionGuard is to leverage the *spatiotemporal inconsistency* property of PAEs to detect anomalies. It predicts the motion states from historical ones and compares them with the current driving states to identify any motion inconsistency caused by physical attacks. We evaluate 9 state-of-the-art PAEs against both camera and camera-LiDAR fusion-based object classification & detection models. Experimental results in both simulation and physical world validate the effectiveness and robustness of VisionGuard. Codes, demo videos and appendix can be found on our anonymous website: https://sites.google.com/view/visionguard.

CCS Concepts

• **Security and privacy → Systems security**.

*Corresponding author

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3670296

Keywords

Defense, Adversarial Attacks, Autonomous Driving

ACM Reference Format:
Xingshuo Han, Haozhao Wang, Kangqiao Zhao, Gelei Deng, Yuan Xu, Hangcheng Liu, Han Qiu, and Tianwei Zhang. 2024. VisionGuard: Secure and Robust Visual Perception of Autonomous Vehicles in Practice. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3658644.3670296

1 Introduction

Autonomous driving technology has been widely commercialized, as evidenced by the increased types of Autonomous Vehicles (AV) transitioning from concepts to real products on public roads. The brain of these vehicles is the Autonomous Driving System (ADS), which incorporates multiple modules to understand the external environment and make safe and accurate driving decisions. One important module is the Visual Perception Module (VPM), which leverages the camera and LiDAR as the primary sensors for perceiving the surrounding context of the vehicle, and then uses state-of-the-art Deep Neural Network (DNN) models for object classification and detection. This VPM lays the foundation of the ADS and plays a critical role in ensuring safe driving.

However, recent studies have shown that VPM is vulnerable to physical adversarial examples (PAEs) [1]. An external adversary can carefully craft adversarial objects to deceive the DNN models to make wrong perception results. Such incorrect results will be further fed into the subsequent modules, leading to incorrect driving decisions and endangering the safety of AVs. PAEs can be constructed to achieve various attack goals, e.g., misclassifying an object [2–4], hiding an existence object [3–9], or recognizing a non-existence object [10–12]. For instance, pasting a malicious sticker onto a stop sign can cause the traffic sign classification model to misrecognize it as a speed limit sign, potentially making the vehicle fail to decelerate at a pedestrian crossing [2]. Similarly, strategically

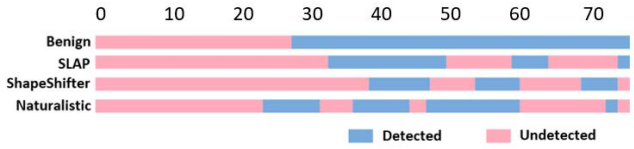


Figure 1: Due to the physical constraints, existing PAEs cannot remain consistency over different frames (from 0_{th} to 70_{th}) when the vehicle approaches them.

placing a 3D-printed obstacle in the middle of a driving lane can deceive the camera-LiDAR fusion-based object detection mechanism, resulting in a collision risk [9].

Driven by the severity of these attacks, numerous methods have been proposed to mitigate PAEs [13]. They can be roughly classified into two categories. (1) *Certified defenses.* They detect adversarial patches with theoretical guarantees of model robustness against white-box attacks. Typical techniques include randomized cropping [14], interval bound propagation [15], de-randomized smoothing [16], secure aggregation [17], feature space masking [18], certified training [19] and objectness explaining [20]. (2) *Vision-based consistency checking.* It leverages the consistent information obtained from different sources to detect PAEs. Such consistency detection can be achieved at the perceptual level [21, 22], physical context level [23–25], or sensor level [26–28].

However, these methods suffer from several limitations in practical scenarios. ① **Limited generalizability.** Existing methods only target specific vision tasks, sensors, or attack goals, as shown in Table 1. Specifically, certified defenses are mainly designed for 2D patch attacks other than 3D LiDAR attacks. They either focus on the classification task [14–19], or object detection task [20], but are unable to cover all the vision tasks in the VPM simultaneously. For vision-based consistency checking, some methods [21, 22] extract and monitor anomalies in motion feature consistency of the target object. They can only detect misclassification attacks, but not object-hiding attacks since there are no targets for feature extraction. ② **Reliance on contextual information.** Many vision-based consistency-checking approaches highly rely on the availability of abundant contextual information from the VPM. For instance, some solutions [23] leverage reasonable relationships between the target object and coexisting benign objects to identify anomalies. They are less effective when there is barely any object other than the target one in the scene. ③ **Computational inefficiency.** Certified defenses normally come with heavy mathematical proofs and algorithms to ensure their robustness. Their high computational cost makes them impractical to achieve real-time protection. More discussions of existing defense works are in Section 2.3.

This paper proposes a practical defense solution to overcome the above limitations. The key insight of our solution is an intriguing phenomenon: *existing PAEs are far from being perfect.* Although a variety of techniques (e.g., EoT [29]) have been applied to enhance their robustness, *it is still infeasible for them to remain consistency against physical variations over time,* as demonstrated in Figure 1. Such phenomenon has been widely studied in prior works [30–32], and we also theoretically and empirically validate this in Section 3.

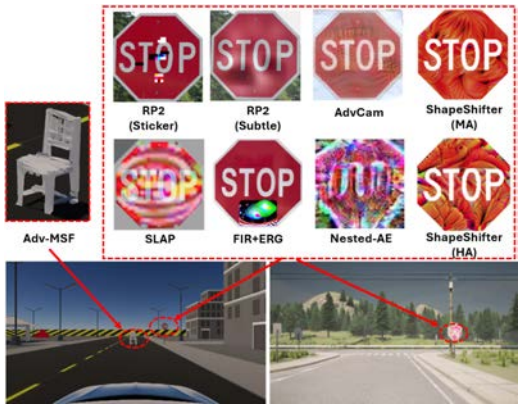


Figure 2: AEs deployed in LGSVL and CARLA.

`VisionGuard` predicts the vehicle's future motion based on its past motion states. If this prediction deviates from the actual movement, it will issue a warning, continuously monitor its movement status, and further determine to eliminate the warning or issue an attack warning based on the cumulative threshold. Specifically, it consists of three modules (Figure 4). (1) State Correction Module (SCM) is responsible for obtaining the vehicle's actual states. It collects the raw IMU and GPU measurements and applies the Kalman Filter algorithm to calibrate the driving state. (2) State Prediction Module (SPM) is used to predict the driving states by tracking historical states. It adopts the Autoregressive Integrated Moving Average algorithm to extract two types of features. (3) Attack Detection Module (ADM) is introduced to assess whether the vehicle is currently safe or under attack. It combines all the predicted states and extracted features to make comprehensive comparisons with a state machine at runtime.

Compared to existing methods, `VisionGuard` has the following advantages: **1) High generalizability**. `VisionGuard` is effective against different forms of PAEs, attack goals, target models and sensors in a holistic manner. **2) Context-free**. `VisionGuard` utilizes a completely different set of data and sensors (i.e., GPS and IMU) from prior works for inconsistency checking and anomaly detection. Such data are stable to provide internal references of the vehicle, and not constrained by external and contextual factors. **3) Efficient computation.** `VisionGuard` captures the motion state of the vehicle with only a few explicit variables, without the need to preprocess high-dimensional image and point cloud inputs. In addition, it predicts states via a lightweight yet effective statistical model, which can satisfy the real-time requirement on resource-constrained platforms.

We comprehensively evaluate `VisionGuard` in industry-grade simulators (LGSVL [33] and CARLA [34]) and physical scenarios. Experiment results show that `VisionGuard` is robustness against 9 state-of-the-art PAEs (Figure 2) in 9 scenarios with different environment conditions (distance, angle, heading, weather, context information). Compared with two representative defense methods based on certification and consistency-checking, `VisionGuard` exhibits the highest detection rate, shortest response time, and best generalization capability. We further conduct a comprehensive analysis of the adaptive attacks and prove that the optimization results

Table 1: Comparison with representative state-of-the-art defense methods. HA: hiding attack; AA: appearing attack; MA: misclassification attack.

Defense Methods		Sensor		Task		Attack Goal		
		Camera	LiDAR	Classification	Detection	HA	AA	MA
Certified Defense	Randomized Cropping [14]	✓		✓				✓
	Interval Bound Propagation [15]	✓		✓				✓
	De-randomized Smoothing [16]	✓		✓				✓
	Secure Aggregation [17]	✓		✓				✓
	PatchGuard++ [18]	✓		✓				✓
	Certified Training [19]	✓		✓				✓
	DetectGuard [20]	✓			✓	✓		
Vision Consistency	PercepGuard [21]	✓		✓	✓		✓	✓
	AdvIT [22]	✓			✓			✓
	SCEME [23]	✓			✓	✓	✓	✓
	SCENE [24]	✓			✓	✓	✓	✓
	KEMLP [25]	✓		✓				✓
	Zhang et al. [26]	✓	✓		✓		✓	
Spatiotemporal Consistency	**VisionGuard (Ours)**	✓	✓	✓	✓	✓	✓	✓

of these hyper-parameters in various scenarios are consistent and reliable. Our contributions are:

- We perform both theoretical analysis and empirical evaluations of 9 state-of-the-arty PAEs in diverse scenarios to show the infeasibility of perfect and robust attacks.
- We propose VisionGuard, the first-of-its-kind method leveraging GPS and IMU measurements to guard the VPM. VisionGuard uses the spatiotemporal inconsistency in the vehicle's kinetic information to detect different types of attacks (object misclassification, hiding or appearing) against different sensors (camera, camera-LiDAR fusion) with different attack techniques.
- VisionGuard is a plug-and-play solution that can work with off-the-shelf ADSs. It is a flexible method with high transferability and scalability.
- VisionGuard achieves high detection rate with low false positives in both simulation and real-world scenarios.

2 Background and Related Work

2.1 Visual Perception Module in ADS

A typical ADS implements a Visual Perception Module (VPM) to understand the environment. The VPM is responsible for processing sensor data and applying DNN models to execute perception tasks, such as object detection, classification, and tracking. Based on the perception outputs, the ADS makes decisions, e.g., throttle, braking, and steering, to ensure the AV operates correctly. In this paper, we focus on the security aspects associated with the VPM. State-of-the-art ADSs typically employ two main approaches for building the VPM: (1) camera-based design, such as Tesla [35] and Intel Mobileye [36]. The ADS relies on 2D or 3D cameras for perception, often using multiple cameras positioned in different locations to improve accuracy and robustness. (2) Camera-LiDAR fusion-based design, such as Baidu Apollo [37] and Google Waymo [38]. The ADS combines both cameras and LiDAR sensors to collect image and 3D point cloud data, respectively. These two modal data are processed separately and then fused to generate the final perception results. This process is normally achieved via a rule-based Multi-Sensor Fusion (MSF) function [37]. We aim to defeat attacks that target both perception designs. We do not consider the LiDAR-only implementation, which has not yet been deployed by the mainstream AV manufacturers in reality.

2.2 Physical Adversarial Examples (PAEs)

Machine learning models are vulnerable to adversarial attacks [2, 3, 3–12], where small-scale perturbations in the input can mislead the victim model to make wrong predictions. Most of these pixel-wise perturbations are nearly imperceptible to human eyes. Despite their stealthiness, many of these attacks utilize the entire input space for perturbation injection. In the physical world, the attacker can achieve such an results by creating localized perturbation in the form of sticker patches, projection patterns, or 3d-printed obstacles. Although such adversarial objects are subject to physical constraints, their threats to ADSs are very severe.

Physical attacks to camera-based perception. Attacks targeting the camera-based VPM can be classified into three categories based on the attack goals. (1) **M**isclassification **A**ttacks (**MA**) deceive the models into classifying the target object into another one [2, 4, 5]. (2) **H**iding **A**ttacks (**HA**) fool the models into ignoring the presence of the target object [3, 5, 39]. (3) **A**ppearing **A**ttacks (**AA**) aim to make the perception model detect a non-existent object [3–9].

Physical attacks to MSF-based perception. Existing works mainly focus on HA. Cao et al. [9] were the first to successfully 3D-print optimized obstacles, such as benches, toy cars, and traffic cones, to deceive camera-LiDAR fusion-based perception. Abdelfattah et al. [8] proposed a similar HA technique within a comparable threat model. However, there is still limited research on robust black-box attacks against MSF-based perception.

2.3 Existing Defenses

Past works have proposed different types of defense solutions to mitigate the above physical attacks. They can be classified into the following two categories.

Certified defenses. Such methods aim to detect adversarial patches, particularly those created by white-box adversaries. Chiang et al.

[15] introduced a certified defense through Interval Bound Propagation (CertIBP), which demonstrates superior robustness against adversarial patches of varying shapes and sizes. Levine et al. [16] extended the robustness of certification with De-Randomized Smoothing (DS) by leveraging the spatially constrained properties of adversarial patches. Similarly, Lin et al. [14] utilized these properties to achieve attack detection by classifying random crops of the input image and generating final outputs based on the majority of the classification results. Metzen et al. [19] improved the efficiency of the certification process by combining it with model training. Several defense strategies are designed specifically to address the localized adversarial patches for CNN models with small receptive fields. For instance, PatchGuard [17] leverages these small receptive fields to limit the impact of corrupted features. It utilizes secure aggregation to retrieve correct prediction results. PatchGuard++ [18] and DetectorGuard [20] are extended over PatchGuard, which applies masks in the feature space to boost the robustness.

Limitations. Certified defenses suffer from several limitations. First, they have scalability and efficiency issues. Their computational cost grows exponentially with the size of inputs, which makes them less practical for real-time applications with high-dimensional inputs, e.g., autonomous driving. Second, while some methods claim to detect physical adversarial patches, their feasibility in physical settings is not well demonstrated. Third, certified defenses are often designed to defend against specific types of attacks, lacking the generalizability and ability to handle diverse attacks.

Vision-based consistency checking. AVs running in the physical world exhibit temporal continuity, which could be disrupted by adversarial attacks. Consistency-based defenses focus on detecting anomalies by utilizing the spatiotemporal information derived from the perceptual, physical, or sensor level. These defenses analyze the consistency of data over time to identify any discrepancies caused by adversarial attacks. (1) Perceptual level. PercepGuard [21] explores the spatiotemporal consistency of the target objects by constantly monitoring its trajectory to detect *MA*. Similarly, AdvIT [22] analyzes the temporal consistency of the target across continuous frames with optical flow estimation of pseudo frames for detection comparison. (2) Physical environment level. Some approaches exploit the consistency properties between the target object and coexisting objects in the scene to detect adversarial attacks. Li et al. [23] created an auto-encoder for each target class to discover whether its contextual discrepancy rules have been violated. Yin et al. [24] employed a language model with an awareness of describing natural scene images to obtain relationships between multiple coexisting objects. Some other researchers assign specific classes with their deterministic attributes. KEMLP [25] combines these attributes with a set of weak auxiliary models to check the consistency properties of the target object. Wang et al. [46] utilized a similar approach to exploit context inconsistencies specifically for persons across different views to detect adversarial attacks. (3) Sensor level. Zhang et al. [26] checked the consistency of data collected from different cameras to detect optical signal attacks by analyzing the distribution of disparity error between them. Xiao et al. [47] leveraged the global and average local differences between normal and adversarial objects to detect *AA* in the point cloud.

Limitations. These methods that rely on external perceptual information have limitations in generalizability. For example, PercepGuard [21] and AdvIT [22] are only effective against *MA* with norm-bounded perturbations. They may not be applicable to *HA* when contextual information is limited. Approaches that rely on the consistency of coexistent objects assume abundant contextual information, which may not be the case in low-light or rural areas. Additionally, some methods [25, 46] focus on verifying the consistency of static objects, which is not applicable to complex spatiotemporal features associated with moving objects.

Defenses for other cyber-physical systems. `VisionGuard` leverages the spatiotemporal inconsistency present in AV's internal kinetic behaviors (i.e., real and predicted motion state) to detect PAEs. We observe the similar strategy has been applied to safegurad other cyber-physical systems, including UAV (Unmanned Aerial Vehicle) [48–50], spacecraft [51–54], etc. Specifically, Chen et al. [48] predicted UAV's future moments based on the LS-SVM algorithm, and detected abnormal states of observation data through residuals. Choi et al. [49] detected external sensor attacks by checking whether the perceived physical state is consistent with the expected state determined by the control model in UAV. Hundman et al. [51] leveraged LSTM to detect anomalies in multivariate time-series metrics of spacecraft based on prediction errors. These methods cannot be directly applied to the protection of AVs, due to the distinct features of the systems and attacks in consideration. For instance, none of the above works target the physical attacks to the visual perception. `VisionGuard` is the first to exploit the kinetic spatiotemporal inconsistency to detect PAEs against visual perception modules in autonomous driving. This is achieved with innovative methods, e.g., integrating and leveraging ARIMA's combination of long-term and short-term predictions to enhance PAE detection.

3 Design Insight

3.1 Threat Model

Attack objective and capability. We consider an external attacker that aims to compromise the VPM of an AV by deploying 2D or 3D PAEs. The attacker can achieve different types of goals introduced in Section 2.2, including *MA*, *HA*, and *AA*. We assume a strong attacker, who knows all the details of the target DNN models in the ADS to generate effective adversarial objects by considering the physical constraints (e.g., naturalness, printability, etc.).

Attack scope. Following the threat model in [21], we do not consider digital adversarial examples. Executing digital attacks requires the attacker to gain access to the Controller Area Network (CAN) bus to inject digital signals, which is more challenging to accomplish in real-time ADSs. We assume the deployed PAE is stationary. This holds true for the mainstream 2D patch and 3D object attacks. Dynamic PAEs (e.g., using screen or projector) for better robustness and concealment are beyond the scope of this work. We concentrate on the optimization-based PAEs with high stealthiness, while ignoring other types of attacks (e.g., phantom attack [39]) that cause significant changes to the environment. Additionally, we mainly focus on the attacks against the vision sensors (i.e., camera, LiDAR), while trusting motion sensors (i.e., GPS, IMU). While motion sensor signals can also be spoofed [55], a plenty of countermeasure have been designed to protect the GPS signals, including authentication,

Table 2: Physical adversarial attacks evaluated on a *real road* from existing works.

Sensor	Task	Method	Transformation	Adversarial Object	Target	Attack Success Rate (%)		Distance (m)			Inconsistent Periods (s)	
						White-box	Black-box	1-5	5-10	10-20	White-box	**Black-box**
Camera	Classification	AdvCam [3]	$\mathcal{S}, \mathcal{R}, \mathcal{B}$.	Patch	Traffic Sign	✗	✗	✗	✗	✗	✗	✗
	Classification	AdvLB [40]	$\mathcal{R}, \mathcal{T}, \mathcal{B}lur$	Laser	Traffic Sign	MA: 77.43	✗	✗	✗	✗	✗	✗
	Classification	ShadowAttack [41]	$\mathcal{S}, \mathcal{B}, \mathcal{PT}, \mathcal{B}ulr$	Shadow	Traffic Sign	MA: 95.91	✗	✗	✗	✗	✗	✗
	Classification	RP2 [42]	$\mathcal{S}, \mathcal{R}, \mathcal{B}$	poster	Traffic Sign	MA: 100 *	✗	✗	✗	✗	0:00-0:03	✗
	Classification	RP2 [42]	$\mathcal{S}, \mathcal{R}, \mathcal{B}$	art	Traffic Sign	MA: 84.8	✗	✗	✗	✗	0:00-0:04	✗
	Detection	ShapeShifter [4]	$\mathcal{S}, \mathcal{R}, \mathcal{T}$	Patch	Traffic Sign	MA: 36-47	✗	✗	✗	✗	0:07-0:09 0:09-0:12 0:13-0:15	All attacks, all 27 videos are discontinuously
	Detection	Nested-AE [10]	–-	Patch	Traffic Sign	AA: 63-81 HA: 60-75	✗	✗	✗	✗	0:13-0:14, 0:21-0:27 0:24-0:27, 0:37-0:39 0:43-0:45	✗
	Detection	Translucent Patch [43]	$\mathcal{S}, \mathcal{T}$	Patch	Traffic Sign	HA: 21.54-42.27	✗	✗	✗	✗	✗	✗
	Detection	Poltergeist [44]	–-	Blurring	Traffic Sign	✗	HA: 98.3 CA: 43.7 AA: 43.1	✗	✗	✗	✗	All videos are discontinuously except HA-TUNNEL**
	Detection	SLAP [5]	$\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{PT}. \mathcal{B}ulr$	Optical	Traffic Sign	HA: 77-89.5	✗	✗	< 5	< 89.5/77.5	5:45-5:48	-
	Detection	TPatch [45]	$\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{RGB}$	Patch	Traffic Sign	MA: 86.4-100	MA: 64.2-85.3	100	85	✗	✗	✗
MSF	Detection	MSF-adv [9]	$\mathcal{R}, \mathcal{T}$	3D obstacle	Cone & Bench	HA: 84.8	✗	✗	✗	✗	✗	Not given

$\mathcal{S}$ Scaling/Shearing $\mathcal{R}$ Rotation $\mathcal{T}$ Translation $\mathcal{B}$ Brightness $\mathcal{PT}$ Perspective Transformation $\mathcal{B}lur$ Gaussian/Motion Blur/Noise $\mathcal{RGB}$ Contrast, Saturation, Hue
✗: We did not get the data/details from their paper. *: The attack success rate shown in their video is not 100%. **: The camera shakes badly.

integrity verification, encryption-based, and deep learning-based techniques [56]. The inherent design features of IMUs can also confer resistance to external spoofing due to their operational independence and lack of reliance on external signals. How to design a defense solution over all untrusted sensors is challenging and never studied in prior works. This will be our future direction.

3.2 Key Insight

The key insight of VisionGuard comes from an observation that **existing physical adversarial examples are not perfect in our real world**. The physical constraints render such attacks less robust and consistent against environmental changes, including light conditions or movement. This conclusion has been confirmed by prior works [30–32]. We also extensively and comprehensively investigate 9 state-of-the-art PAEs, as summarised in Table 2. We download all the videos from these papers and create a database (can be found on our website). We carefully review these videos and observe that none of these attacks can be consistent over time, as shown in the "Inconsistent Period" column of Table 2.

We argue that **it is infeasible to create *perfect* physical adversarial examples *in practice***. Below we provide our justifications from two perspectives.

3.3 Difficulty of Generating Perfect PAEs

We first theoretically demonstrate that it is difficult for a stationary PAE to achieve ideal attack results in a dynamic AD scenario. Specifically, we prove that, as a camera mounted on an AV constantly approaches a PAE from far to near, it captures a multitude of video frames, each presenting a slightly different perspective. This continuous change makes it extremely hard to maintain a high probability that a universal PAE is effective across all these rapidly increasing and varying frames.

Theorem 3.1. *Consider a case where the object x is successfully attacked with the attacking probability to be $P(F(x+\delta) \neq y) = 1 - p_{x+\delta}$ where δ denotes the perturbation. Further, we consider that the image of each frame contains $d \times d$ pixels. Then, the maximum probability $P_{\max}$ of successfully attacking all frames is exponentially reduced with the increasing number N of the total frames, i.e.,*

$$P_{\max} \leq (1 - p_{x+\delta})^{\frac{2N}{d}\left(\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1\right)}, \tag{1}$$

where $\bar{d} \times \bar{d}$ represents the number of object pixels at the location when the vehicle starts to see the object.

Proof. For clarity of reading, we here mainly present the sketch of proof. More details can be found in Appendix B. To establish the proof, we view the problem of attacking object detection to be a binary classification task where the object is either detected or not. We denote y to be the label where the object is recognized and y' otherwise. Besides, we denote $P(F(x) = y) = p_x, P(F(x+\delta) = y) = p_{x+\delta}$ for ease of analyzing. Also, we use $l(x, y)$ to represent the loss of classifying the example x to be y. Obviously, we have $l(x, y) < l(x, y')$ for any clean example x. By denoting $f(x) = \frac{l(x+\delta, y') - l(x,y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x,y)\|}$, we have

$$\begin{aligned} &P(F(x+\delta) = y) = P\big(l(x+\delta, y) < l(x+\delta, y')\big) \\ \geq &P\Big(l(x,y) + \nabla_x l(x,y)^T \delta + \frac{L\epsilon^2}{2} < l(x+\delta, y')\Big) \\ \Rightarrow &P\Big(\frac{\nabla_x l(x,y)^T \delta}{\|\nabla_x l(x,y)\|} < f(x)\Big) < p_{x+\delta}, \end{aligned} \tag{2}$$

where the first inequality is derived by applying the Taylor expansion over $l(x+\delta, y)$. Considering the perturbation budget $\|\delta\| \leq \epsilon$, we can compute the expectation of the $\frac{\nabla_x l(x,y)^T \delta}{\|\nabla_x l(x,y)\|}$ as

$$\mathbb{E}\frac{\nabla_x l(x,y)^T \delta}{\|\nabla_x l(x,y)\|} \leq p_{x+\delta} f(x) + (1 - p_{x+\delta})\epsilon. \tag{3}$$

Instead of considering the vehicle coming closer to the object, we consider driving the vehicle back from the position where it is parallel to the object for ease of analysis. We use $g(x+\delta) = x + \gamma\delta$ to denote the transformed example of $x + \delta$ where $\gamma < 1$ is the

scaled size. Based on the L-smoothness assumption, we have

$$l(x,y) + \gamma \nabla_x l(x,y)^T \delta - \frac{L\gamma^2\epsilon^2}{2} \leq l(g(x+\delta),y) \leq l(x,y) + \gamma \nabla_x l(x,y)^T \delta + \frac{L\gamma^2\epsilon^2}{2}. \quad (4)$$

Similarly, based on (4), we have

$$P(F(g(x+\delta)) \neq y) \leq \frac{\gamma\left(p_{x+\delta} \cdot \left(l(x+\delta,y') - l(x,y) - \frac{L\epsilon^2}{2}\right) + (1-p_{x+\delta})\epsilon\|\nabla_x l(x,y)\|\right)}{l(g(x+\delta),y') - l(x,y) - \frac{L\gamma^2\epsilon^2}{2}} \quad (5)$$

Considering the vehicle stays in the position where the scale size s satisfies $\gamma \leq \frac{1-p_{x+\delta}}{2-3p_{x+\delta}}$, then bringing the scale size of s back to (35) derives the probability bound of $P(F(g(x+\delta)) \neq y)$:

$$P(F(g(x+\delta)) \neq y) \leq 1 - p_{x+\delta}. \quad (6)$$

Further, denoting the number of total frames as N and the size of each frame as $d \times d$, we can obtain that each captured picture corresponds to $2N/d$ frames. By further denoting the minimum scale size corresponding to recognizing clean figures is $\bar{d}$, we can calculate the maximum scale times r of which the attacking probability is less than $1 - p_{x+\delta}$ as $r = \lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor$. We can then compute the number n of frames of which the attacking probability is less than $1 - p_{x+\delta}$ as

$$n = \frac{2N(r+1)}{d} = \frac{2N}{d}\left(\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1\right). \quad (7)$$

As a consequence, by denoting the probability of successfully attacking one frame as P_a, the maximum probability of successfully attacking all frames is less than the following probability:

$$P_{\max} = P_a^N \leq (1-p_{x+\delta})^n = (1-p_{x+\delta})^{\frac{2N}{d}\left(\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1\right)},$$

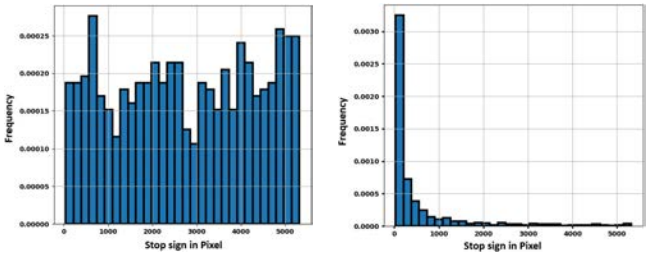
which completes the proof. □

From the right hand of inequality (1), Theorem 3.1 reveals that the probability of successfully attacking all frames is exponentially reduced with the number of frames. Taking the case in Figure 1 as an example where there is a total of 70 frames to be attacked, the probability of successfully attacking all of these frames is extremely small. As a consequence, it is natural that only partial frames are attacked successfully and PAEs cannot remain consistent over time.

In the above analysis we consider a general scenario on object detection, which is the target of most state-of-the-art attacks in AD contexts. The decision in object detection is regarded as binary classification in which the object is detected or not. It is important to note that our proof also applies to the misclassification scenario: once the model prediction is induced to the target categories, e.g., RP2 attacks [42] cause the misclassification of a stop sign as deceleration or going straight, it becomes a binary classification task.

3.4 Ineffectiveness of Robustness-enhanced Solutions

To overcome the physical constraints and make the PAE more robust, many attacks leverage some robustness-enhanced techniques,



(a) Assumed distribution in EoT. (b) Real world distribution.

Figure 3: Distributions of stop sign size in 416*416 images.

i.e., Expectation of Transformation (EoT) or its variants, which augment the training of PAEs with uniform transformation. We show that these techniques are still incapable of generating perfectly consistent PAEs from two perspectives.

(1) Unreasonable assumption. It is a common practice in previous studies to uniformly sample the transformation operation within a certain range [4, 5, 45, 57]. However, Wang et al. [58] demonstrated that this assumption is inconsistent with the physical model. To further verify this, we experiment with a stop sign in the real world. First, we drive a physical vehicle in uniform motion from 50 meters away towards a standard stop sign (600mm diameter), where the Intel RealSense D435i camera with 1920 * 1080 resolution is mounted on the vehicle. We record the video and obtain the stop sign pixel for each frame, for a total of 1470 frames. The stop sign distributions assumed in EoT used in [57] and in the real world are given in Figure 3. We observe that in reality, the vast majority of pixel sizes lie in the range 0 to 200 (small pixel distribution when the AD vehicle is far away from the object). However, [57] implies that when the object is small, attack convergence becomes challenging, making it difficult to attack. As a consequence, uniformly sampling object sizes without consideration of their real-world frequency can lead to less robust PAEs. It might be overly robust to size variations that rarely occur (large pixel distribution) and underprepared for common size variations (small pixel distribution), making the PAE easily affected by distance.

(2) Ignored context. EOT involves adding random distortions during the optimization process to make the perturbation more robust. However, it primarily focuses on the PAE itself, without sufficiently considering the background context, which is critical for AD scenarios. In the real world, the detection model not only analyzes the object of interest in isolation, but also consider its context, including the background and the relationship between the object and its surrounding environment. Thus the background context can significantly impact the effectiveness of PAEs [10]. As vehicles move from far to near, the surrounding environment of the PAE changes quite significantly, yet EOT does not take these background changes into account. Although [10] considers background context and improves EoT, its robustness is still poor due to the limitations posed by the dynamic changes during driving.

3.5 Design Overview

Such imperfection of physical attacks provides a new opportunity for mitigating them. In this paper, we exploit the inconsistency in spatiotemporal motion features of vehicles to detect PAEs. Specifically, motion characteristics include a vehicle's global position,

speed, acceleration, and heading to describe its behavior in space and time. When a vehicle is in motion, its spatiotemporal feature is continuously changing due to continuous variation of actuator behavior. When the vehicle's movement is spatialtemporal consistent, this signifies that the vehicle is currently doing a smooth and regular motion variation. On the contrary, when the vehicle's spatialtemporal feature pattern exhibits sudden and irregular change, its motion state tends to be spatiotemporal inconsistent.

In a standard ADS platform, the perception results will be transmitted to the subsequent planning and control modules for real-time on-road driving, so the spatiotemporal inconsistencies introduced by PAEs can directly affect the behaviors of the vehicle, including its position, heading, velocity, and acceleration, which are continuously updated while the vehicle is in motion. Therefore, by monitoring these kinetic variables, it is feasible to detect any potential anomalies that may arise as a result of PAEs.

We check the inconsistency of motions instead of the perception results [59–63] for the following reasons. (1) Different categories of objects exhibit unique features in different perception models. If we focus on checking consistency properties in the perception results, we need to build separate detection models for different combinations of object categories and models, which is infeasible for complex traffic scenarios. (2) The motion state and physical properties of a vehicle are more consistent and predictable since the throttle and braking levels defined by the control module under different scenarios are deterministic. To measure the vehicle's internal state, IMU and GPS data are also more robust and reliable against extreme environmental conditions compared with cameras and LiDARs. As long as the control flow in the ADS is secure, the vehicle's reactions in benign environments can be uniformly modeled with higher precision.

4 VisionGuard

We present VisionGuard, a practical motion-based defense framework against various PAEs to AVs. It is designed as a plug-and-play module to conduct safety checking at each decision-making stage of the ADS operation pipeline, without the necessity of modifying existing functional modules or interfering with their intermediate results. VisionGuard is fundamentally different from other methods in the following aspects: (1) Compared to existing certified and digital-level defenses [14–17, 19, 20, 64], VisionGuard is more general and computationally efficient as it does not rely on theoretical robustness on specific models and extensive computations over large-scale inputs. (2) Compared to vision-based and perception-level consistency checking defenses [21–23, 25, 46], VisionGuard focuses on the internal state consistency of the vehicle, which can exclude the effects of external factors and contextual information.

Figure 4 overviews VisionGuard, which mainly contains three modules: State Correction Module (SCM), State Prediction Module (SPM), and Attack Detection Module (ADM). Specifically, **(1)** SCM collects the GPS and IMU data to obtain the vehicle's current state. However, the state information can contain errors or inconsistencies due to various dynamic effects caused by the environment and power train. To address this issue, SCM applies Kalman Filter (KF) [65] to get the corrected driving state S_t^{KF} at time t. The state serves two main purposes. First, it is the basis for consistency checking with the predicted driving states in ADM. Second, it is stored in SPM to predict the next state of the vehicle. **(2)** SPM is designed for estimating the driving states. It leverages Autoregressive Integrated Moving Average (ARIMA) to build a kinetic model, which utilizes a set of stored historical states collected from SCM, to produce two types of estimated states: a long-term predicted state S_t^{LT} and a short-term predicted state S_t^{ST} at time t. **(3)** Subsequently, the corrected state S_t^{KF}, as well as the estimated states S_t^{LT} and S_t^{ST} are forwarded to ADM for consistency checking. ADM implements a safety state machine to regulate vehicle operations. It performs calculations to determine two residual values $||S_t^{KF} - S_t^{ST}||$ and $||S_t^{KF} - S_t^{LT}||$. If both values exceed their respective thresholds (w_1, w_2), ADM initiates the safety state to "*Warning*" for a predefined time interval l. Concurrently, it begins tracking the Accumulated State Prediction Residual (AR) of the state prediction residuals. If this value exceeds a predefined threshold (h) in l, it will set to "*Attack*" to signalize the occurrence of physical attacks and make a safe operation, e.g., point brake. Otherwise, it will go back to the "*Normal*" state.

4.1 State Correction Module (SCM)

The motion states of the vehicle, e.g., position, heading, velocity, and altitude, are provided by onboard sensors like GPS and IMU. However, raw data from these sensors are noisy and collected at varied rates, making them unsuitable for anomaly detection without appropriate calibration. To bridge this gap, SCM applies Kalman Filter (KF) to obtain a corrected state S_t^{KF}. KF is a popular method for achieving corrected state estimation in robotic vehicles by combining outputs from diverse sensors. By utilizing KF, we can approximate the motion state of the vehicle within a small time interval and provide a reasonable estimation of the vehicle's behavior. Therefore, we provide the certified motion state of the vehicle from a kinematic perspective.

Formally, the vehicle's motion state s_t at time t includes its heading position $\overrightarrow{p} = \overrightarrow{(x, y, \theta)}$ and velocity $\overrightarrow{v}$, which can be expressed as: $s_t = \left[\overrightarrow{p}, \overrightarrow{v}\right]^T$. For computation efficiency and motion state optimization, SCM assumes that the vehicle is conducting a uniformly accelerated rectilinear motion (constant vehicle throttle force) during each running timestep. Based on this premise, we can obtain a theoretical state prediction $s_{t|t-1}$ of the current timestamp through KF prediction of the previous timestamp $S_{t-1|t-1}^{KF}$, state transition matrix F, control matrix B and control vector u_t from vehicle's current acceleration a_t (see Appendix A.1 for detailed derivation):

$$s_{t|t-1} = \begin{bmatrix} 1 & 0 & \Delta t \cos\theta & 0 \\ 0 & 1 & \Delta t \sin\theta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ v_{t-1} \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\cos\theta\Delta t^2 \\ \frac{1}{2}\Delta\sin\theta\Delta t^2 \\ \Delta t \\ 0 \end{bmatrix} \times a_t \quad (8)$$

$$s_{t|t-1} = F \cdot S_{t-1|t-1}^{KF} + Bu_t \quad (9)$$

Every KF prediction step combines raw measurement z_t and theoretical prediction $s_{t|t-1}$ of the current timestamp with Kalman Gain K and observation matrix H, which incorporate the relative influence of the measurement and the prediction in the state estimation process. Therefore, the final prediction output of SCM can

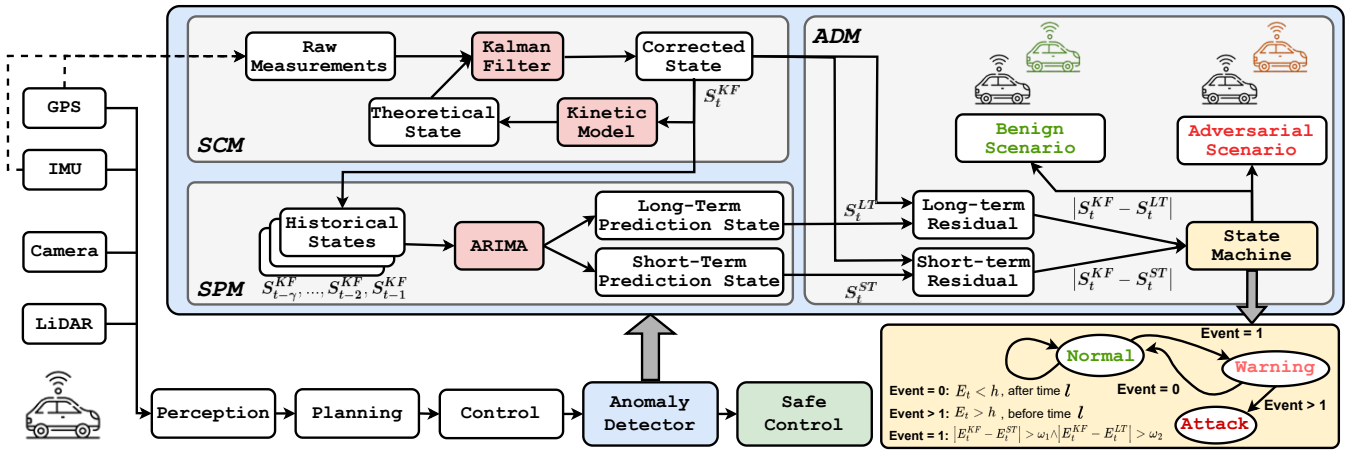


Figure 4: Overview of VisionGuard. The solid lines represent the data flows in VisionGuard and the big grey arrows zoom into the detailed internal structures of Anomaly Detector and State Machine modules.

be expressed as:

$$S_{t|t}^{KF} = K_t z_t + (I - K_t H) s_{t|t-1} \tag{10}$$

By leveraging KF estimation, SCM can output a more accurate prediction of the motion state by considering the presence of noises and errors in the raw sensor measurements. Overall, it combines the predicted states based on physical dynamics with the measured states into one corrected motion state variable, thus providing an accurate vehicle state for subsequent consistency checking.

4.2 State Prediction Module (SPM)

This module leverages Auto-Regressive Integrated Moving Average (ARIMA) to predict the vehicle's current state S_t^{ARIMA} based on the historical records $\{S_{t-\gamma}^{KF}, ..., S_{t-2}^{KF}, S_{t-1}^{KF}\}$ collected from SCM. ARIMA is a statistical model widely used for time series prediction. It can analyze historical time-sequence data and generate reliable forecasts of future values. The ARIMA model comprises two key components: (1) the Auto-Regressive (AR) model captures the relationship between the current state S_t^{ARIMA} and its historical states; (2) the Moving Average (MA) model accounts for the accumulated errors in the auto-regressive process. By combining these two components, ARIMA leverages the temporal dependencies and noise patterns present in the historical data to ensure robust and accurate prediction of the vehicle's non-periodic motion behaviors. It is formally defined as follows:

$$S_t^{ARIMA} = \underbrace{\mu + \sum_{i=1}^{p} \phi_i S_{t-i}^{ARIMA}}_{\text{AR}} + \epsilon_t + \underbrace{\sum_{i=1}^{q} \theta_i \epsilon_{t-1}}_{\text{MA}} \tag{11}$$

where μ is an intercept constant, and ϵ_t is the error term (usually represents the white noise) at timestamp t. The parameters ϕ_i and θ_i are specific to the AR and MA components and optimized during the training stage.

ARIMA first offers computational efficiency advantages over methods based on recurrent neural networks (RNN), which have

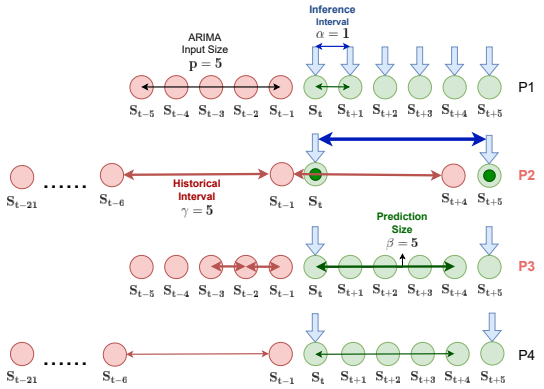


Figure 5: ARIMA model inference process. Red: SPM inputs; Green: SPM outputs; Blue: ARIMA inference; P2: ST state prediction; P3: LT state prediction.

complex deep network architectures that require weight optimization and extensive training on large-scale datasets. Second, ARIMA prioritizes gaining insights into recent motion trends, patterns and providing efficient and interpretable results, rather than only focusing on achieving the highest prediction accuracy.

Key hyper-parameters in SPM. Figure 5 shows some ARIMA model inference processes. The input size p of the model defines how far back in time the model looks for making new predictions. It is an important parameter, which can affect model's prediction outcomes. Additionally, we introduce three key hyper-parameters, denoted as SPM (α, β, γ), to offer a more comprehensive prediction of vehicle's motion state. (1) *SPM inference interval α:* this refers to the length of timesteps between two consecutive ARIMA inferences at runtime. Frequent ARIMA inference enables faster anomaly detection. However, it requires SPM to achieve a high prediction accuracy in benign scenarios to prevent false alarms from being frequently triggered. Besides, it also results in increased computational costs. (2) *SPM prediction size β:* this refers to the number of motion states for prediction during one ARIMA inference. While

Algorithm 1 Attack Detection

Input: ARIMA inference parameters p, α, β, γ; State vectors received at frame i from raw sensor measurements: S_i^z; State vector predictions from Kalman Filter: S_{i+1}^{KF}, Sort-term: S_{i+1}^{ST} and Long-term: S_{i+1}^{LT}; Accumulated residual at frame $i+1$: AR_{i+1}; Alarm thresholds for Short-term prediction: ω_1, Long-term prediction: ω_2 and Accumulated residual: h; Caution interval: l; **Output:** *Normal* or *Attack*

$AR \leftarrow 0$, *Normal*
while running time steps t exceeds $\alpha * \gamma$ **do**
  $S_{t+1}^{KF} \leftarrow KF(S_t^z, S_t^{KF})$
  if $t\%a = 0$ **then**
    $S_{t+1}^{ST} \leftarrow ARIMA(S_{t-(p-1)*\gamma}^{KF}, S_{t-(p-2)*\gamma}^{KF}, ..., S_t^{KF})$
    $S_{t+1,t+2,...t+\beta}^{LT} \leftarrow ARIMA(S_{t-(p-1)}^{KF}, S_{t-(p-2)}^{KF}, ..., S_t^{LT})$
  end if
  if $||S_{t+1}^{KF} - S_{t+1}^{ST}|| > w_1$ and $||S_{t+1}^{KF} - S_{t+1}^{LT}|| > w_2$ **then**
    Warning mode for l steps
  end if
end while

//*Warning* mode
while in *Warning* mode **do**
  $AR_{t+1} \leftarrow AR_t + ||S_{t+1}^{KF} - S_{t+1}^{ST}|| + ||S_{t+1}^{KF} - S_{t+1}^{LT}||$
  if $AR_{t+1} > h$ **then**
    Raise *Attack*
    Break
  end if
end while
$AR \leftarrow 0$, *Normal*

the ARIMA model is often used to predict a single value per step (P1 in Figure 5), they can also be utilized to forecast a variation tendency with a set of multiple values. SPM defines a trade-off between capturing motion trends and maintaining prediction accuracy. As ARIMA predicts more values, it increasingly relies on previous predictions rather than the true values to make new predictions. Once the number of predicted values reaches the input size p, future predictions will be made entirely based on historical predictions. It is important to carefully balance this trade-off for VisionGuard to generalize well across different types of attacks. (3) *SPM historical interval γ:* this defines how far back in the historical states should be included as inputs for each ARIMA inference. Since the input size p of the model is fixed, we can increase γ to take further historical data into consideration without having to increase p. Due to the inconsistency property of these attacks, including more historical state information indicates a larger deviation of SPM predictions from SCM estimations in adversarial cases, which can benefit attack detection. However, in more complex benign scenarios where the vehicle does not consistently accelerate, decelerate, or maintain a steady speed, including outdated historical state information can adversely impact the SPM's ability to make accurate benign predictions. Additionally, storing historical data for an extended period will lead to increased system memory consumption.

Prediction Types. We introduce two state prediction types within SPM: (1) *Short-Term State Prediction* (ST). This employs a longer historical data interval to make predictions for a short-term future. It aims to capture more extensive state variation trends and provide predictions that reflect the recent short-term behavior of the vehicle (P2 in Figure 5). (2) *Long-Term State Prediction* (LT). This utilizes a shorter historical data interval to make predictions for a long-term future. It aims to capture the immediate state variation trend and provide predictions that depict the general long-term behavior of the vehicle (P3 in Figure 5). Introducing these two inference types can strike a balance between accuracy and tendency in the obtained predictions. This allows us to effectively capture both the short-term and long-term motion behaviors of the vehicle. Combining the outputs of LT (S_t^{LT}) and ST (S_t^{ST}) enables VisionGuard to make comprehensive assessments that generalize well in a variety of adversarial scenarios.

4.3 Attack Detection Module (ADM)

The corrected state S_t^{KF} from SCM, with the predicted states S_t^{ST}, S_t^{LT} from SPM, are fed into the ADM module in real-time. Algorithm 1 shows how ADM performs consistency checking. We introduce three important variables in ADM: Long-Term State Prediction Residual (LTR), Short-Term State Prediction Residual (STR), and Accumulated State Prediction Residual (AR). LTR and STR are utilized to quantify the difference between the real and predicted states of the vehicle, which are defined as $||S_t^{KF} - S_t^{LT}||_2$ and $||S_t^{KF} - S_t^{ST}||_2$, respectively. Detailed derivation can be found in Appendix A.2. AR is defined as the accumulated value of LTR and STR in the subsequent timestamps, which is expressed as:

$$AR = \sum_{t=t_w}^{t_w+l} (||S_t^{KF} - S_t^{LT}||_2 + ||S_t^{KF} - S_t^{ST}||_2) \tag{12}$$

where t_w is the timestamp when "warning" is triggered. We set specific thresholds (w_1, w_2, h) as indicators for detecting deviations and anomalies in the vehicle's motion state. When LTR and STR simultaneously surpass their respective thresholds w_1 and w_2, ADM will initiate a *Warning* state and be alert to potential threats. Note that a sudden change in LTR and STR does not necessarily indicate the presence of an attack. In benign scenarios, various sudden state inconsistencies frequently occur due to factors such as accelerating from a static velocity, braking for red lights or pedestrians, and other normal driving maneuvers. Therefore, rather than raising an *Attack* state directly, ADM presents a *Warning* state for a specific time interval l. During *Warning*, ADM will continuously monitor AR to determine if it has exceeded h. If yes, ADM will trigger an *Attack* state to signalize the occurrence of an attack. If no further exceedance is observed within l, ADM will switch the vehicle from the *Warning* state to the *Normal* state, and reset AR to 0. These precautionary measures ensure that ADM gains a comprehensive understanding of the cumulative deviations in motion states over time before raising an alarm. ADM relies on the values of hyper-parameters w_1, w_2, h, and l to make accurate decisions. Properly defining these hyper-parameters is crucial to effectively counter false positives. We will discuss this in Section 7.2.

5 Simulation Evaluation

We validate the effectiveness of each module in VisionGuard, followed by the end-to-end AD simulator evaluation.

Experiment Setup. We use LGSVL and CARLA to conduct experiments. Our evaluation encompasses various environmental factors, including diverse weather conditions, traffic scenarios, road types, etc. The frequencies of IMU and GPS equipped on driving vehicle are 100Hz and 12.5Hz, respectively. For each scenario, both benign and adversarial cases are tested independently. At runtime, raw sensory data are processed by different DNN models in the perception module. The perception results are subsequently published

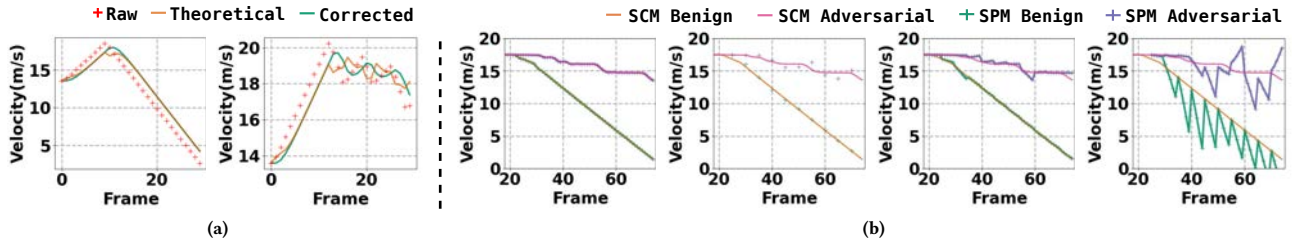


Figure 6: (a) Benign and adversarial. (b) SPM outputs with different configurations.

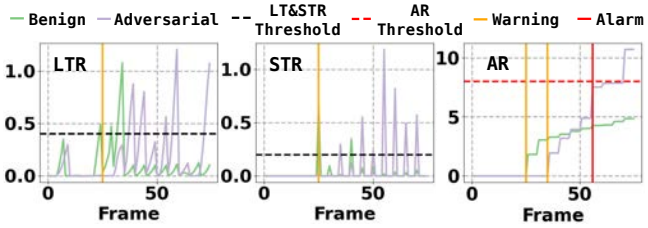


Figure 7: LTR (left), STR (middle) and AR (right) in ADM (w_1: 0.2, w_2: 0.4, h: 8, l: 50).

to the simulator API to generate control commands. The server configuration is given in Appendix A.3. We mainly use Yolo-v3 as the victim model and hiding attack based on the SLAP technique [5], if not mentioned specifically.

Metrics. We adopt the following metrics to evaluate `VisionGuard`. (1) Detection Success Rate (DR): this is defined as the ratio of adversarial scenarios that can be successfully detected. (2) False Positive Rate (FPR): this is the portion of benign cases that are incorrectly detected as adversarial cases out of all the benign cases. (3) Short-Term Excessive Rate (STER) and Long-Term Excessive Rate (LTER): they are calculated as the ratio of frames that exceed the respective thresholds. These metrics provide insights into how stable the attack is during the entire run from short-term and long-term perspectives individually. (4) Accumulated State Prediction Residual (AR): this is calculated by summing up all the values of residuals generated by both types of motion states. It is considered as the key evaluation metric in ADM and provides a comprehensive assessment of physical attacks.

5.1 Evaluation of SCM

SCM collects the raw measurements from GPS and IMU at each timestamp, to produce the corrected motion state of the vehicle. Table 13 in Appendix shows the detailed configurations in LGSVL, where the positions and control variables are indicated based on the PythonAPI [66] of LGSVL. Figure 6(a) presents the raw measurements, theoretical and corrected states of the vehicle in benign and adversarial scenarios. We have the following observations. First, in the benign case, the vehicle accurately detects a stop sign and exhibits stable deceleration until coming to a stop. However, in the adversarial case, the vehicle fails to maintain a consistent deceleration due to unstable detection outputs. Consequently, the vehicle fails to stop in front of the stop sign, indicating that the attack has been successfully launched. Second, SCM effectively minimizes the errors between the theoretical states and raw measurements. One key advantage of integrating SCM into `VisionGuard` is the smoothing effect it imparts on the runtime state outputs, which is particularly noticeable in the speed fluctuations of the vehicle. This reduction in variation plays a crucial role in minimizing the potential margin of errors, especially in cases where abrupt measurement errors can potentially mislead the outputs of `VisionGuard`.

5.2 Evaluation of SPM

To train the ARIMA model, we generate a dataset by driving the vehicle in the map and recording the SCM outputs. The vehicle follows a predefined sequence of actions, including gradually accelerating from a stationary position, maintaining a consistent speed for a specific duration, steering in different directions, and uniformly decelerating to a stop. We collect a total of 100 scenarios.

After the dataset is obtained, we train the ARIMA model and retrieve its corresponding weights. To showcase the impact of the inference interval α, prediction size β, and historical interval γ, we conduct a series of motion state predictions and compare the outputs of SPM and SCM. All experiments are carried out in the same scenario, where the vehicle is driving at a constant speed and encounters both a benign and adversarial stop sign. Table 4 presents four different SPM hyper-parameter configurations (P1, P2, P3, P4) to predict the future velocity of a vehicle, which can be summarised as: **P1**: use short-term historical data from five previous consecutive frames to make a state prediction at every step. **P2**: use long-term historical data by extracting estimations across 25 previous frames with a five-frame interval to make a state prediction at every five steps. **P3**: use short-term historical data from five previous consecutive frames to make five consecutive state predictions at every five steps. **P4**: use long-term historical data by extracting estimations across 25 previous frames with a five-frame interval to make five consecutive state predictions at every five steps.

Table 4: Different inference settings in SPM.

	p	α	β	γ
P1	5	1	1	1
P2	5	5	1	5
P3	5	5	5	1
P4	5	5	5	5

Figure 6(b) shows the experimental results with the configurations P1-P4. We observe that with P1, the SPM predictions exhibit slight deviations from SCM estimations for both benign and adversarial scenarios, while they differ significantly from P4. This highlights the crucial trade-off between prediction accuracy and tendency in SPM, as discussed in Section 4.2. Both configurations represent extreme cases where the balance between the two factors is heavily skewed, resulting in predictions that are

Table 3: Detection rates (DR, %) of different PAEs ($w_1 : 0.2, w_2 : 0.4, h : 10$).

Attacks	Attack Goal	Target object	Victim Model	STER	LTER	AR	*Warning* (frame)	*Attack* (frame)	DR
SLAP	HA	Stop sign	YOLO-v3	0.06	0.16	16.817	Yes(35)	Yes(65)	100%
RP2	MA	Stop sign	Faster R-CNN	0.08	0.17	17.269	Yes(40)	Yes(65)	100%
ShapeShifter	MA	Stop sign	Faster R-CNN	0.12	0.16	15.160	Yes(35)	Yes(65)	100%
ShapeShifter	HA	Stop sign	Faster R-CNN	0.12	0.17	17.143	Yes(25)	Yes(50)	100%
AdvCam	MA	Stop sign	VGG-19	0.06	0.05	11.900	Yes(45)	Yes(50)	100%
Nested-AE	AA	Stop sign	Faster RCNN	0.12	0.24	15.545	Yes(25)	Yes(35)	100%
Adv-MSF	HA	Bench	Apollo-v5	0.1	0.16	14.589	Yes(35)	Yes(60)	100%

Table 5: Comparisons with two baseline defenses.

Method	DR		Detection Latency (s)		Runtime (per frame)
	SLAP	ShapeShifter	SLAP	ShapeShifter	
DetectGuard	**71.1%**	0	7.22	N/A	117.2ms
PercepGuard	0	**100%**	N/A	6.73	32ms
VisionGuard	**100%**	**100%**	**6.54**	**5.03**	**3.1ms**

indistinguishable between benign and adversarial scenarios. SPM enables the vehicle to capture the underlying benign motion trend while minimizing the loss of prediction accuracy, which is achieved by extending historical knowledge (P2) or predicting the future states (P3).

5.3 Evaluation of ADM

The corrected states generated by SCM and the prediction states generated by SPM are fed into ADM. Figure 7 illustrates the mechanism of ADM at runtime. In a benign scenario, we observe that the vehicle encounters a sudden increase in both STR and LTR, surpassing their respective thresholds in the 25th frame. This indicates that the current prediction results from SPM exhibit a relatively strong deviation from SCM's corrected estimation, which is caused by commands sent from the perception module to the control module in response to an event that triggers a variation in the detection results, such as the appearance of a stop sign in front. However, it is important to note that this "stop sign" could potentially be an adversarial object crafted by an attacker to mislead the perception models. Hence, ADM promptly triggers a *Warning* state (Event = 1 in the state machine), starting to record and monitor the value of AR. After a short time interval ($l = 50$ steps), if there is no significant further increase in AR, the warning state will be disabled, and AR is reset to zero (Event = 0 in the state machine). The vehicle then resumes safe driving until a new *Warning* is triggered.

In adversarial scenarios, the *Warning* state is also triggered but relatively more slowly (35th frame). Different from the benign scenario, the values of S^{ST} and S^{LT} continue to experience significant fluctuations in the warning state. Hence, the *Attack* state is triggered in the 65th frame as AR also surpasses the threshold, forcing the vehicle to disregard any subsequent commands and initiate safe control measures immediately, e.g., gradual deceleration to a stop.

5.4 End-to-end Evaluation of VisionGuard

We comprehensively evaluate VisionGuard by using 9 different state-of-the-art PAEs with various target objects, models, perception functions, and attack goals, as well as 9 different scenarios. This allows us to collect a total number of 36000 frames from 36 runs to fully validate VisionGuard. **Defense Effectiveness.** For each attack, we run 10 times in LGSVL. We observe all the AEs can be 100% detected in the benign sunny condition. We also consider other scenarios such as rainy or foggy conditions to check the false positives, and the details are given in Section 7.2. Table 3 reports the DR of different attacks. It is clear that VisionGuard has successfully detected all types of adversarial objects in different settings.

Table 6: Runtime analysis for one detector iteration.

	Mean
Kalman Filtering	0.000012s
ARIMA(short-term)	0.000612s
ARIMA(long-term)	0.000684s
Detection	0.000007s
Total	0.001315s

Comparison with Baseline Methods. We compare VisionGuard with representative open-source defense methods including DetectGuard [20] (Certified defense) and PercepGurad [21] (Vision-based consistency checking). PercepGuard is originally designed for defending car-to-person misclassification attacks, for a fair comparison, we use speed limit 50 as the induced label, and yolov3 is fine-tuned on German Traffic Sign Recognition Benchmark dataset [67]. We record 45 runs with adversarial objects created by SLAP [5] and ShapeShifter [4] in 5 different conditions, 3 different initial speeds, object headings, and maps to see if these defense methods can successfully detect the adversarial patches attached to the stop sign. A full run lasts 10s with 10 fps. Notably, for the two defense methods, once a single or a period of frames during the adversarial run is deemed as being attacked, the run is considered to be detected successfully. In such settings, the effectiveness of VisionGuard shown in Table 5 is evident, as it successfully detects the adversarial patches in every run. We also observe DetectGuard and PercepGuard also achieve high DRs, among which, DetectGuard cannot achieve 100% because it is designed for localized patch hiding attacks, while the AE generated by SLAP is a universal patch. Moreover, these two methods are designed to address specific attack goals, VisionGuard is capable of detecting physical attacks across all different goals. This makes VisionGuard a comprehensive solution for attack detection. Furthermore, for detection latency and runtime (per frame), VisionGuard outperforms the other methods, achieving faster average detection speeds and less runtime overhead for both types of attacks. Notably, PercepGuard requires 4-5 frames of inference to detect attacks, so its average detection time is around 128-160ms. Table 6 gives the detailed runtime overhead of each module in VisionGuard. This indicates that VisionGuard is not only effective but also efficient, providing timely protection.

Defense Robustness. We comprehensively evaluate the robustness of VisionGuard against 9 different scenarios. The detailed description of these scenarios can be seen in Appendix A.3. We run each evaluation 10 times. Table 7 shows the average DR over 90 evaluation scenarios. We observe that VisionGuard can achieve 100% detection rates on average, which shows a high robustness of

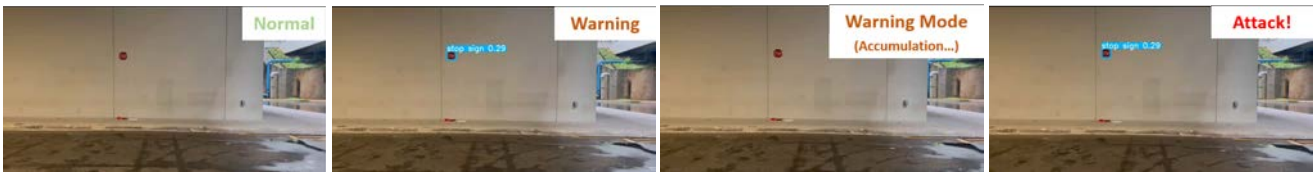
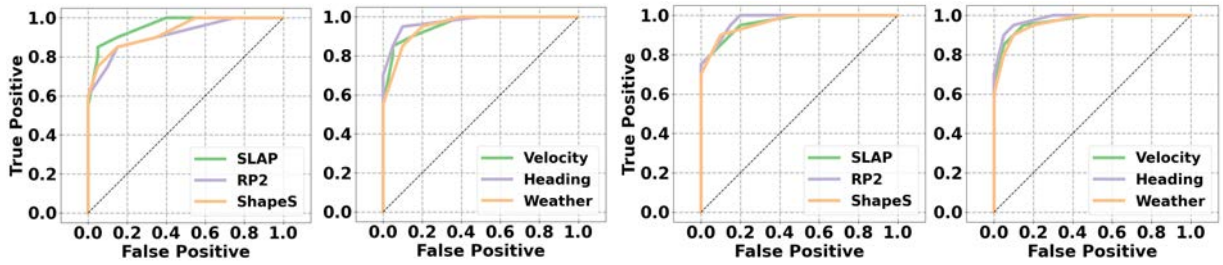


Figure 8: Physical experiment results. *Normal, Warning, Warning mode, Attack* happen at 0-34, 35, 35-39, 40 (frame), respectively.

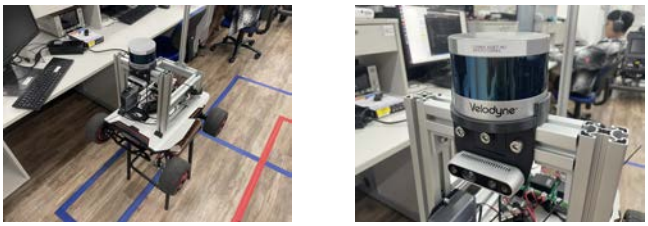


(a) h **(**w_1**: 0.2,** w_2**: 0.4,** l**: 50)** **(b)** w_2 **(**w_1**: 0.2,** h**: 10,** l**: 50)**

Figure 9: ROC of AR threshold h (a) and w_2 (b).

Table 7: Detection rates (DR, %) in 9 different scenes.

Scene	SLAP		RP2		ShapeShifter		DR
	AR	*Attack* (frame)	AR	*Attack* (frame)	AR	*Attack* (frame)	
Default	16.82	Yes(65)	17.27	Yes(65)	17.14	Yes(50)	100%
Heading	13.23	Yes(55)	14.02	Yes(55)	12.85	Yes(60)	100%
Contextual	19.67	Yes(60)	12.46	Yes(60)	15.49	Yes(60)	100%
Twilight	18.10	Yes(65)	19.10	Yes(65)	18.94	Yes(60)	100%
Rainy	18.96	Yes(65)	18.27	Yes(60)	20.32	Yes(60)	100%
Foggy	17.42	Yes(65)	18.03	Yes(60)	19.65	Yes(60)	100%
Road	15.20	Yes(55)	16.58	Yes(60)	14.08	Yes(55)	100%
Velocity	15.40	Yes(35)	13.83	Yes(40)	16.44	Yes(40)	100%
Map	16.07	Yes(55)	15.36	Yes(55)	13.18	Yes(55)	100%



(a) UGV **(b) Front-facing camera**

Figure 10: Our physical UGV with an Intel RealSense D435i camera and Velodyne-16 LiDAR.

`VisionGuard` against different attack approaches under different scenarios. While our method demonstrates strong defense performance overall, we also observe instances where it produces false positives in certain scenarios. False positives occur when `VisionGuard` incorrectly identifies benign inputs as adversarial objects. This can potentially lead to unnecessary disruptions in normal ADS operations. We give details in Section 7.2.

6 Outdoor Road Driving Test

Setup. The experiments are carried out on a closed campus road using an Unmanned Ground Vehicle (UGV), as depicted in Figure 10. The UGV is equipped with an Intel RealSense D435i front-facing camera (1920×1080 resolution), and a Bosch BMI055 6-axis IMU through which we obtain the relative positioning information. We use our High-definition (HD) map to provide positioning information since the UGV does not have a GPS device installed. We believe using GPS could yield the same results. Unlike the camera set at 10 fps in simulations, the physical camera on the UGV is 30 fps, thus giving a more accurate indication of how effective `VisionGuard` can be at different frame rates in the real world. In our experiments, both benign and adversarial stop signs (HA, generated by [5]) are color-printed and placed at 1.8m height. The UGV is initially placed 10 meters away from the target sign and drives straight forward. We set the initial speed as 5m/s (18km/h) at the beginning of each recording. Such a setup is significant since it models the real traffic environment. We repeat the experiment for 30 runs with each run consisting of two benign and adversarial signs heading towards different angles.

Table 8: Detection rates (DR, %) and False Positive Rates (FPR, %) with different horizontal distances in real-world.

	STER	LTER	AR	*Attack*	DR	FPR
Benign/left-1.5m	0.18	0.95	1.53	No	N/A	0%
Benign/right-1.5m	0.31	1.06	2.74	Yes(40)	N/A	3.3%
Adversarial/left-1.5m	0.88	1.45	7.13	Yes(40)	100%	N/A
Adversarial/right-1.5m	0.84	1.61	8.29	Yes(40)	100%	N/A

Results. Table 8 shows the detection results in the physical environment. We observe that the most benign stop sign does not trigger the *Attack* signal. However, all the adversarial patches with different angles can achieve a 100% detection rate for all 30 runs. Figure 8 visualizes the detection results, where the stop sign is placed on the right side road. As the UGV moves towards the adversarial object, the confidence scores for the "stop sign" present distinct inconsistency at varying distances and angles between them. Similarly, the experimental results in the physical world show identical performance as the simulation ones, which provides further evidence for the effectiveness of `VisionGuard`.

Table 9: Number of scenarios and detection rates (DR, %) in pedestrian interaction scenarios.

	1 pedestrian	2 pedestrians
Bright	3 (100%)	4 (100%)
Dim	5 (100%)	3 (100%)

More complex scenarios. To demonstrate the practicality of VisionGuard, we further simulate more complex scenarios on a closed campus road involving interactions with pedestrians, considering the government restrictions on physical attack testing in the public environment. Specifically, we collect 15 scenarios featuring 1-2 pedestrians under 2 lighting conditions (as shown in Table 9), with pedestrians walking around while the UGV drives towards the PAE. VisionGuard can still achieve 100% detection rates in these scenarios. Experiment videos can be found in our project website.

7 Discussion and Limitation

7.1 Adaptive Attack

7.1.1 Mechanism-aware adaptive attack. We consider a more sophisticated attacker who may try to improve his attack techniques in response to our defense measures. The most potential directions is improving the attack robustness. The attacker may invest efforts into improving the robustness of his attack methods to bypass VisionGuard. He may employ advanced algorithms and optimization techniques to generate more powerful adversarial objects that are capable of consistently evading our defense mechanism. Although this is an attractive goal, to the best of our knowledge, there is still no satisfactory way to generate absolutely robust optimization-based PAEs that can overcome the physical constraints and maintain the attack effectiveness consistently.

7.1.2 Parameter-aware adaptive attack. VisionGuard includes some hyper-parameters (e.g., AR, LTR, caution interval l), whose values are determined empirically. We investigate the impact of these hyper-parameters for adaptive attack analysis.

AR and LTR thresholds. Intuitively, if the attacker manages to constantly bypass the setting of h, the *Attack* state would not be triggered during the caution interval l. On the other hand, if the physical attack is robust enough to stay under-covered below w_2, the *Warning* state would not be triggered at all from the start. VisionGuard will not work as a consequence. To thoroughly understand the impact of h and w_2, we conduct extensive experiments to evaluate their performance under three types of adversarial objects and scenarios following the same setups in Tables 3 and 7. For each evaluation, we record the detection False Positives (FP) and True Positives (TP) over a total of 40 runs (20 benign and 20 adversarial cases). To evaluate AR, we fix (w_1, w_2, l) and slide h from 0 to 15 with the interval of 0.05. Similarly for LTR, we fix (w_1, h, l) and slide w_2 from 0 to 1 with the interval of 0.05.

We determine the optimal values of h and w_2 closest to (0,1) for each type of PAEs and scenario. Figures 9 gives the results. It is clearly that the differences between these values across different attack types and scenarios are quite small. This indicates that (1) VisionGuard exhibits high stability and versatility, eliminating the need for employing different thresholds for different attack types and scenarios. (2) In adversarial cases, a slight decrease in the thresholds has a negligible impact on FPs while significantly improving TPs. Table 12 shows the optimal values of LTR and AR with the corresponding AUC, respectively.

Table 10: Attack results with different caution intervals in urban and highway environments.

v_0 (m/s)	l	Benign	Adversarial	DR (%)	FPR (%)
45	80	20	19	95	0
	100	20	20	100	0
	120	18	20	100	10
90	30	20	19	95	0
	50	20	20	100	0
	70	19	20	100	5

Figure 11: Stop sign can be detected in light rain, while a FP case occurs in heavy rain.

Figure 12: Stop sign can be detected in light fog, while a FP case occurs in dense fog.

Caution interval l. We conduct the same experiments on SLAP [5] with two different vehicle initial speeds: *45 km/h* and *90 km/h*, corresponding to the speed limits in urban and highway environments. We observe from Table 10 a negative correlation between l and v_0 in our test range. When the vehicle is running at a lower speed with a more stable motion state variance, it is necessary to increase the size l to capture more information about the consistency pattern in order to reduce the chances of missing detection.

Limitation. In this paper, we do not add real-time non-optimization based PAEs such as controllable projections and signals into our threat model. For instance, the attacker may use a drone to project an patch onto a billboard or other surfaces visible to the ADS's camera. However, these attacks can be effective to some extent. Suppose that the attacker has knowledge of the value of the caution interval l, he would try to project the AE again after the "Caution mode" is deactivated.

7.2 Evaluation Under Normal Cases

Prior experiments prove the robustness of VisionGuard in different environments. However, during our simulation and physical tests, we also noticed that VisionGuard is sensitive to corner cases,

Figure 13: A FP case occurs due to very bumpy road.

resulting in false positives. (indeed, we find not only in their videos but also in our experiments, that DNN models show high robustness of object detection and can easily keep consistent over time).

Simulation. We find adverse weather conditions such as heavy rain and fog (Figures 11 and 12) can cause the perception module to make mistakes and lead `VisionGuard` to falsely trigger the *Attack* alarm. Regarding this case, we demonstrate that FPs primarily occur under extreme weather conditions, while `VisionGuard` is still effective in normal rainy or foggy scenarios, as illustrated in Table 7. Besides, to mitigate the effect of such corner cases, numerous fog-removal and rain-removal approaches [68–73] can be used to enhance the perception module and reduce FPs.

Physical. To fully understand `VisionGuard`'s FP performance in the real world, we collected 30 videos of different scenarios and tested them using Yolov3 and Yolov5, Faster-RCNN [74]. All the videos are given on our website. We observe that these models only fail one case, i.e., image blur due to bumpy roads. Blurry camera images can cause the inaccurate detection of obstacles or traffic signs (Figure 13). Regarding this case, for minor bumps, the integrated IMU within the camera typically performs motion compensation, thereby minimizing vibrations and reducing image blurring. It is important to note that IMU compensation fails only under highly turbulent conditions, as shown in Figure 13, resulting in the false triggering of the *Attack* alarm. Based on the above analysis, our approach shows strong effectiveness and robustness, as well as very low false positives. ADS practitioners should develop more robust perception algorithms in different environments to eliminate the false detection rate of `VisionGuard`.

8 Conclusion

We present `VisionGuard`, a practical defense framework that can effectively detect PAEss by exploiting the spatiotemporal inconsistency in a vehicle's kinetic states. `VisionGuard` is agnostic to the attack goals, target objects, models, sensors, and contextual information. We comprehensively evaluate 9 state-of-the-art PAEs in both simulation and real-world scenarios. Our results show that `VisionGuard` achieves high detection rates across diverse settings with high robustness.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported in part by Nanyang Technological University (NTU)-DESAY SV Research Program under Grant 2018-0980, National Research Foundation, Singapore and DSO National Laboratories under its AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), and National Natural Science Foundation of China under grants 62302184.

References

[1] 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.

[2] 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.

[3] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1000–1008.

[4] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 52–68.

[5] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "{SLAP}: Improving physical adversarial examples with {Short-Lived} adversarial perturbations," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1865–1882.

[6] Y. Zhu, C. Miao, T. Zheng, F. Hajiaghajani, L. Su, and C. Qiao, "Can we use arbitrary objects to attack lidar perception in autonomous driving?" in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1945–1960.

[7] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun, "Physically realizable adversarial examples for lidar object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 716–13 725.

[8] M. Abdelfattah, K. Yuan, Z. J. Wang, and R. Ward, "Towards universal physical attacks on cascaded camera-lidar 3d object detection models," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3592–3596.

[9] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li, "Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," in *SP*, 2021.

[10] 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*, 2019, pp. 1989–2004.

[11] 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 *CCS*, 2019.

[12] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *USENIX Security*, 2020.

[13] H. Ren, T. Huang, and H. Yan, "Adversarial examples: attacks and defenses in the physical world," *International Journal of Machine Learning and Cybernetics*, pp. 1–12, 2021.

[14] W.-Y. Lin, F. Sheikholeslami, L. Rice, J. Z. Kolter *et al.*, "Certified robustness against physically-realizable patch attack via randomized cropping," 2021.

[15] P.-y. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studer, and T. Goldstein, "Certified defenses for adversarial patches," *arXiv preprint arXiv:2003.06693*, 2020.

[16] A. Levine and S. Feizi, "(de) randomized smoothing for certifiable defense against patch attacks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6465–6475, 2020.

[17] 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, pp. 2237–2254.

[18] C. Xiang and P. Mittal, "Patchguard++: Efficient provable attack detection against adversarial patches," *arXiv preprint arXiv:2104.12609*, 2021.

[19] J. H. Metzen and M. Yatsura, "Efficient certified defenses against patch attacks on image classifiers," *arXiv preprint arXiv:2102.04154*, 2021.

[20] C. Xiang and P. Mittal, "Detectorguard: Provably securing object detectors against localized patch hiding attacks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3177–3196.

[21] Y. Man, R. Muller, M. Li, Z. B. Celik, and R. Gerdes, "That person moves like a car: Misclassification attack detection for autonomous systems using spatiotemporal consistency," in *USENIX Security Symposium*, 2023.

[22] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3968–3977.

[23] 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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 396–413.

[24] 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 *proceedings of the IEEE/CVF international conference on*

computer vision, 2021, pp. 7858–7867.
[25] 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*. PMLR, 2021, pp. 3976–3987.
[26] 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*, vol. 8, no. 2, pp. 1140–1153, 2020.
[27] X. Han, Y. Zhou, K. Chen, H. Qiu, M. Qiu, Y. Liu, and T. Zhang, "Ads-lead: Lifelong anomaly detection in autonomous driving systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1039–1051, 2022.
[28] X. Han, K. Chen, Y. Zhou, M. Qiu, C. Fan, Y. Liu, and T. Zhang, "A unified anomaly detection methodology for lane-following of autonomous driving systems," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications*. IEEE, 2021, pp. 836–844.
[29] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 284–293.
[30] N. Hingun, C. Sitawarin, J. Li, and D. Wagner, "Reap: A large-scale realistic adversarial patch benchmark," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4640–4651.
[31] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "No need to worry about adversarial examples in object detection in autonomous vehicles," *arXiv preprint arXiv:1707.03501*, 2017.
[32] P. A. Sava, J.-P. Schulze, P. Sperl, and K. Böttinger, "Assessing the impact of transformations on physical adversarial attacks," in *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, 2022, pp. 79–90.
[33] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," *arXiv preprint arXiv:2005.03778*, 2020.
[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.
[35] Tesla, "Autopilot," https://www.tesla.com/autopilot.
[36] Intel, "Mobileye," https://www.mobileye.com/.
[37] Baidu, "Apollo," https://github.com/ApolloAuto/apollo.
[38] Google, "Waymo," https://waymo.com/.
[39] B. Nassi, D. Nassi, R. Ben-Netanel, Y. Mirsky, O. Drokin, and Y. Elovici, "Phantom of the adas: Phantom attacks on driver-assistance systems," *Cryptology ePrint Archive*, 2020.
[40] R. Duan, X. Mao, A. K. Qin, Y. Chen, S. Ye, Y. He, and Y. Yang, "Adversarial laser beam: Effective physical-world attack to dnns in a blink," in *CVPR*, 2021.
[41] Y. Zhong, X. Liu, D. Zhai, J. Jiang, and X. Ji, "Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon," in *CVPR*, 2022.
[42] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *CVPR*, 2018.
[43] A. Zolfi, M. Kravchik, Y. Elovici, and A. Shabtai, "The translucent patch: A physical and universal attack on object detectors," in *CVPR*, 2021.
[44] 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 *SP*, 2021.
[45] W. Zhu, X. Ji, Y. Cheng, S. Zhang, and W. Xu, "Tpatch: A triggered physical adversarial patch."
[46] 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 *ICCV*, 2021.
[47] Q. Xiao, X. Pan, Y. Lu, M. Zhang, J. Dai, and M. Yang, "Exorcising"wraith": Protecting lidar-based object detector in automated driving system from appearing attacks," *arXiv preprint arXiv:2303.09731*, 2023.
[48] Y. Chen, B. Wang, W. Liu, and D. Liu, "On-line and non-invasive anomaly detection system for unmanned aerial vehicle," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*. IEEE, 2017, pp. 1–7.
[49] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *CCS*, 2018.
[50] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "Savior: Securing autonomous vehicles with robust physical invariants," in *Usenix Security*, 2020.
[51] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.
[52] M. Gorbett, H. Shirazi, and I. Ray, "Sparse binary transformers for multivariate time series modeling," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 544–556.
[53] C. Xiao, Z. Gou, W. Tai, K. Zhang, and F. Zhou, "Imputation-based time-series anomaly detection with conditional weight-incremental diffusion models," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2742–2751.
[54] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, "Dcdetector: Dual attention contrastive representation learning for time series anomaly detection," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3033–3045.
[55] 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 *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 931–948.
[56] Y. He, J. Li, and J. Liu, "Research on gnss ins & gnss/ins integrated navigation method for autonomous vehicles: A survey," *IEEE Access*, 2023.
[57] W. Jia, Z. Lu, H. Zhang, Z. Liu, J. Wang, and G. Qu, "Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems," *arXiv preprint arXiv:2201.06192*, 2022.
[58] N. Wang, Y. Luo, T. Sato, K. Xu, and Q. A. Chen, "Does physical adversarial example really matter to autonomous driving? towards system-level effect of adversarial object evasion attack," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4412–4423.
[59] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.
[60] S. Jung, J. Lee, D. Gwak, S. Choi, and J. Choo, "Standardized max logits: A simple yet effective approach for identifying unexpected road obstacles in urban-scene segmentation," in *ICCV*, 2021.
[61] F. Heidecker, A. Hannan, M. Bieshaar, and B. Sick, "Towards corner case detection by modeling the uncertainty of instance segmentation networks," in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*. Springer, 2021, pp. 361–374.
[62] X. Du, Z. Wang, M. Cai, and Y. Li, "Vos: Learning what you don't know by virtual outlier synthesis," *arXiv preprint arXiv:2202.01197*, 2022.
[63] C. Zhang, Z. Huang, M. H. Ang, and D. Rus, "Lidar degradation quantification for autonomous driving in rain," in *IROS*, 2021.
[64] H. Liu, J. Jia, and N. Z. Gong, "Pointguard: Provably robust 3d point cloud classification," in *CVPR*, 2021.
[65] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.
[66] lgsvl, "Pythonapi," https://github.com/lgsvl/PythonAPI.
[67] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *IJCNN*, 2011.
[68] H. Wang, Q. Xie, Q. Zhao, and D. Meng, "A model-driven deep neural network for single image rain removal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
[69] X. Chen, H. Li, M. Li, and J. Pan, "Learning a sparse transformer network for effective image deraining," in *CVPR*, 2023.
[70] W.-T. Chen, Z.-K. Huang, C.-C. Tsai, H.-H. Yang, J.-J. Ding, and S.-Y. Kuo, "Learning multiple adverse weather removal via two-stage knowledge learning and multi-contrastive regularization: Toward a unified model," in *CVPR*, 2022.
[71] J. M. J. Valanarasu, R. Yasarla, and V. M. Patel, "Transweather: Transformer-based restoration of images degraded by adverse weather conditions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2353–2363.
[72] Y. Wang, C. Ma, and J. Liu, "Smartassign: Learning a smart knowledge assignment strategy for deraining and desnowing," in *CVPR*, 2023.
[73] Y. Zhu, T. Wang, X. Fu, X. Yang, X. Guo, J. Dai, Y. Qiao, and X. Hu, "Learning weather-general and weather-specific features for image restoration under multiple adverse weather conditions," in *CVPR*, 2023.
[74] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

A Artifact Appendix

The artifact evaluation descriptions can be found on our website: https://sites.google.com/view/visionguard.

A Appendix A

A.1 KF derivation

Formally, the state variables include vehicle's position (x and y coordinate) and velocity (v). Vehicle's state S_t at time step t can be expressed as:

$$S_t = [x, y, v]^T \tag{13}$$

We assume that during every time interval, the vehicle is doing a uniformly accelerated rectilinear motion (a) along x and y axis with a heading angle of θ. According to the basic law of kinematics, we can retrieve a theoretical prediction of vehicle's position and velocity:

$$x_t = x_{t-1} + v_{t-1}\Delta t \cos\theta + \frac{1}{2} a \cos\theta \Delta t^2 \tag{14}$$

$$y_t = y_{t-1} + v_{t-1}\Delta t \sin\theta + \frac{1}{2} a \sin\theta \Delta t^2 \tag{15}$$

$$v_t = v_{t-1} + a\Delta t \tag{16}$$

Which in matrix form can be expressed as:

$$S_{t|t-1} = \begin{bmatrix} 1 & 0 & \Delta t \cos\theta & 0 \\ 0 & 1 & \Delta t \sin\theta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ v_{t-1} \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\cos\theta\Delta t^2 \\ \frac{1}{2}\Delta \sin\theta \Delta t^2 \\ \Delta t \\ 0 \end{bmatrix} \times a_t \tag{17}$$

As we define F, B, u_t, s_t respectively as state transition matrix, control matrix, control vector and true state. Eq. 5 can be also written in a compact form as eq. 6. Note here that $S_{t|t-1}$ represents the theoretical state prediction of the current time step and $S^{KF}_{t-1|t-1}$ represents the KF correction state of previous time step. In the follow-up derivation, other variables will also be represented in such forms.

$$S_{t|t-1} = F \cdot S^{KF}_{t-1|t-1} + Bu_t \tag{18}$$

However, true state of vehicle s_t contain noises (ω_t) which we assume that they are independently Gaussian distributed with a covariance matrix of Q_t ($Q_t = cov(\omega_t)$). We can also represent this difference between theoretical state and true state with a covariance matrix $P_{t|t-1}$:

$$S_t = F \cdot S_{t-1} + Bu_t + \omega_t \tag{19}$$

$$P_{t|t-1} = cov(S_t - S_{t|t-1}) = FP_{t-1|t-1}F^T + Q_t \tag{20}$$

Similarly, for raw measurements (z_t), noises can also be expressed in covariance form with a measurement matrix H: R_t and U_t

$$U_t = HP_{t|t-1}H^T + R_t \tag{21}$$

Combining the value of theoretical predicted state ($S_{t|t-1}$) and raw measurements (z_t) through a linear weight (Kalman Gain K), the final KF-corrected result of the current step can be expressed as:

$$s^{KF}_{t|t} = K_t z_t + (I - K_t H) s_{t|t-1} \tag{22}$$

Lastly, Kalman Gain (K) and covariance error (P) are updated through a Least Squares optimization:

Table 11: Training runtime setup.

Map	SingleLaneRoad
Vehicle Initial Position	(-122, 0, -1.9)
Vehicle Initial Rotation	(0, 90, 0)
Vehicle Initial Speed	0
Speed Limit	50
Runtime Duration (s)	10
Runtime FPS	10
Throttle Factor	1
Breaking Factor	1

$$K_t = P_{t|t-1}H^T {U_t}^{-1} \tag{23}$$

$$P_{t|t} = P_{t|t-1}(I - K_t H) \tag{24}$$

A.2 SPM short-term state prediction residual (STR) and long-term state prediction residual (LTR)

In a general ARIMA prediction function: q denotes as the number of lagged forecast errors in the prediction equation. μ denotes as ARIMA intercept constant and ϵ_t denotes as white noise at timestep t. r_i and θ_i respectively denotes as AR and MA specific weights for optimization in the training stage.

Short-term state prediction (S^{ST}_t) In the SPM short-term inference stage, γ denotes as the historical interval (sparity of historical states that SPM utilized to make one single state prediction):

$$S^{ST}_t = \mu + \sum_{i=1}^{\gamma} r_i S^{AR}_{t-\gamma*i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-\gamma} \tag{25}$$

Long-term state prediction ($S^{ST}_t, S^{ST}_{t+1}, ..., S^{ST}_{t+\beta+1}$) In the long-term inference stage, β denotes as the prediction size (the amount of future states that SPM predicts for a single inference step):

$$S^{LT}_t = \mu + \sum_{i=1}^{\beta} r_i S^{AR}_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-1} \tag{26}$$

$$S^{LT}_{t+1} = \mu + \sum_{i=1}^{\beta-1} r_i S^{AR}_{t-i} + r_i S^{LT}_t + \epsilon_{t+1} + \sum_{i=1}^{q} \theta_i \epsilon_t \tag{27}$$

$$S^{LT}_{t+2} = \mu + \sum_{i=1}^{\beta-2} r_i S^{AR}_{t-i} + \sum_{i=1}^{2} r_i S^{LT}_{t+i-1} + \epsilon_{t+2} + \sum_{i=1}^{q} \theta_i \epsilon_{t+1} \tag{28}$$

$$\vdots$$

$$S^{LT}_{t+\beta} = \mu + r_i S^{AR}_t + \sum_{i=1}^{\beta} r_i S^{LT}_{t+i-1} + \epsilon_{t+\beta} + \sum_{i=1}^{q} \theta_i \epsilon_{t+\beta-1} \tag{29}$$

$$S^{LT}_{t+\beta+1} = \mu + \sum_{i=1}^{\beta} r_i S^{LT}_{t+i-1} + \epsilon_{t+\beta+1} + \sum_{i=1}^{q} \theta_i \epsilon_{t+\beta} \tag{30}$$

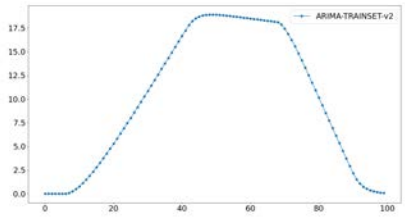


Figure 14: ARIMA training set.

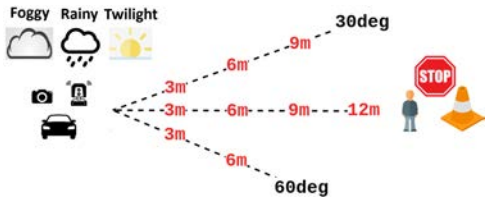


Figure 15: External environment variables in our setting.

Figure 16: Different simulation scenes.

Table 12: Optimal value of LTR threshold.

		SLAP	RP2	ShapeShifter	Velocity	Heading	Weather
LTR	**LTR**	0.44	0.44	0.44	0.44	0.39	0.44
	AUC	0.85	0.89	0.88	0.93	0.95	0.93
AR	**AR**	0.68	0.73	0.73	0.73	0.68	0.73
	AUC	0.89	0.86	0.88	0.90	0.93	0.90

A.3 Additional evaluation results

ARIMA training set. Training set setup and motion state data collected during a single run are shown in Table 11 and Figure 14.

Runtime analysis. Table 6 shows the runtime performance of VisionGuard on a single image (800*600, 32-bit color) with a PC equipped with a AMD Ryzen 9 5900X 12-Core Processor and one Nvidia GeForce RTX 3080 GPU.

Scenario description. (1) Heading: changing the heading direction of the adversarial object by 30° towards the vehicle. (2) Contextual: adding a person and another vehicle close to the adversarial object. (3) Twilight: removing sunlight and adding lamp light. (4) Rainy: adding raindrops. (5) Foggy: adding foggy effects. (6) Road condition: adding damage and wetness effects on the ground. (7) Initial velocity: changing the initial velocity from 0 to 5. (8) Map: adding "SanFrancisco" map from the LGSVL. Figure 16 visualizes some demo settings.

Table 13: SCM runtime setup.

Map	CubeTown
Stop Sign Position	(12.24, 0, 7.56)
Stop Sign Rotation	(0, 92.86, 0)
Adversarial Example	SLAP[5]
Vehicle Initial Position	(-3.2, 0, -35)
Vehicle Initial Rotation	(0, 180, 0)
Vehicle Initial Speed	14
Speed Limit	20
Runtime Duration (s)	3
Runtime FPS	10
Objectness Threshold	0.25
Throttle Factor	1
Breaking Factor	1

B Appendix B: Proof for Theorem 3.1

To establish the proof, we view the problem of attacking object detection to be a binary classification task where the object is either detected or not. We denote y to be the label where the object is recognized and y' otherwise. Besides, we denote $P(F(x) = y) = p_x, P(F(x+\delta) = y) = p_{x+\delta}$ for ease of analyzing. Also, we use $l(x, y)$ to represent the loss of classifying the example x to be y. Obviously, we have $l(x, y) < l(x, y')$ for any clean example x. By denoting $f(x) = \frac{l(x+\delta,y') - l(x,y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x,y)\|}$, we have

$$\begin{aligned}
&P(F(x+\delta) = y) = P\big(l(x+\delta, y) < l(x+\delta, y')\big) \\
\geq &P\Big(l(x,y) + \nabla_x l(x,y)^T\delta + \frac{L\epsilon^2}{2} < l(x+\delta, y')\Big) \\
= &P\Big(\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|} < \frac{l(x+\delta,y') - l(x,y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x,y)\|}\Big) \\
= &P\Big(\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|} < f(x)\Big) \\
\Rightarrow &P\Big(\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|} < f(x)\Big) < p_{x+\delta},
\end{aligned} \tag{31}$$

where the first inequality is derived by applying the Taylor expansion over $l(x+\delta, y)$. Considering the perturbation budget $\|\delta\| \leq \epsilon$, we can compute the expectation of the $\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|}$ as

$$\mathbb{E}\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|} \leq p_{x+\delta} f(x) + (1 - p_{x+\delta})\epsilon. \tag{32}$$

Instead of considering the vehicle coming closer to the object, we consider driving the vehicle back from the position where it is parallel to the object for ease of analysis. We use $g(x+\delta) = x + \gamma\delta$ to denote the transformed example of $x+\delta$ where $\gamma < 1$ is the scaled size. Based on the L-smoothness assumption, we have

$$\begin{aligned}
&l(x,y) + \gamma\nabla_x l(x,y)^T\delta - \frac{L\gamma^2\epsilon^2}{2} \\
&\leq l(g(x+\delta), y) \leq l(x,y) + \gamma\nabla_x l(x,y)^T\delta + \frac{L\gamma^2\epsilon^2}{2}.
\end{aligned} \tag{33}$$

Based on (33), we have

$$\begin{aligned}
&P(F(g(x+\delta))=y)\\
=&P\big(l(g(x+\delta),y)<l(g(x+\delta),y')\big)\\
\geq&P\Big(l(x,y)+\gamma\nabla_x l(x,y)^T\delta+\frac{L\gamma^2\epsilon^2}{2}<l(g(x+\delta),y')\Big)\\
=&P\Big(\gamma\nabla_x l(x,y)^T\delta<l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}\Big)\\
=&P\Big(\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|}<\frac{l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}}{\gamma\|\nabla_x l(x,y)\|}\Big)\\
\geq&1-\frac{\mathbb{E}\frac{\nabla_x l(x,y)^T\delta}{\|\nabla_x l(x,y)\|}}{\frac{l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}}{\gamma\|\nabla_x l(x,y)\|}}\\
\geq&1-\frac{\gamma(p_{x+\delta}f(x)+(1-p_{x+\delta})\epsilon)\|\nabla_x l(x,y)\|}{l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}}.
\end{aligned}\tag{34}$$

Since $P(F(g(x+\delta))=y)=1-P(F(g(x+\delta))\neq y)$, we have

$$\begin{aligned}
&P(F(g(x+\delta))\neq y)\\
<&\frac{\gamma\Big(p_{x+\delta}f(x)+(1-p_{x+\delta})\epsilon\Big)\|\nabla_x l(x,y)\|}{l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}}\\
=&\frac{\gamma\Big(p_{x+\delta}\cdot\Big(l(x+\delta,y')-l(x,y)-\frac{L\epsilon^2}{2}\Big)+(1-p_{x+\delta})\epsilon\|\nabla_x l(x,y)\|\Big)}{l(g(x+\delta),y')-l(x,y)-\frac{L\gamma^2\epsilon^2}{2}}
\end{aligned}\tag{35}$$

Considering the probability is positive, we can conclude that the denominator and numerator have the same sign. For simplicity, we consider $l(x+\delta,y')\geq l(x,y)$ and both of them to be positive. Note that the following derivation is similar when they are negative, which derives the same conclusion. Specifically, we aim to find the value of the scale size s when the probability $P(F(g(x+\delta)\neq y))\leq 1-p_{x+\delta}$:

$$\begin{aligned}
&P(F(g(x+\delta)\neq y))\leq 1-p_{x+\delta}\Leftrightarrow\gamma p_{x+\delta}l(x+\delta,y')\\
&-l(g(x+\delta),y')(1-p_{x+\delta})+\gamma\epsilon(1-p_{x+\delta})\|\nabla_x l(x,y)\|\\
&+l(x,y)(1-p_{x+\delta}-\gamma p_{x+\delta})\leq\big(\gamma p_{x+\delta}+\gamma^2p_{x+\delta}-\gamma^2\big)\frac{L\epsilon^2}{2}.
\end{aligned}\tag{36}$$

Applying Taylor expansion of $l(g(x+\delta),y')$ over (37) derives

$$\begin{aligned}
&\gamma p_{x+\delta}l(x+\delta,y')-(1-p_{x+\delta})\Big(l(x+\delta,y')\\
&+\nabla_{x+\delta}l(x+\delta,y')^T(g(x+\delta)-(x+\delta))+\frac{L(g(x+\delta)-(x+\delta))^2}{2}\Big)\\
&+l(x,y)(1-p_{x+\delta}-\gamma p_{x+\delta})+\gamma\epsilon(1-p_{x+\delta})\|\nabla_x l(x,y)\|\\
&\leq\big(\gamma p_{x+\delta}+\gamma^2p_{x+\delta}-\gamma^2\big)\frac{L\epsilon^2}{2}\\
&\underset{(a)}{\Leftrightarrow}(\gamma p_{x+\delta}+p_{x+\delta}-1)(l(x+\delta,y')-l(x,y))\\
&\leq\big(\gamma p_{x+\delta}+\gamma^2p_{x+\delta}-\gamma^2+(1-p_{x+\delta})(\gamma-1)^2\big)\frac{L\epsilon^2}{2}
\end{aligned}\tag{37}$$

where (a) holds as considering $\nabla_x l(x,y)\approx 0$ and $\nabla_{x+\delta}l(x+\delta,y')\approx 0$. Since $\gamma\leq 1$ and $p_{x+\delta}<\frac{1}{2}$, we have $\gamma p_{x+\delta}+p_{x+\delta}-1<0$. Thus, the formula (37) holds when

$$\begin{aligned}
&\gamma p_{x+\delta}+\gamma^2p_{x+\delta}-\gamma^2+(1-p_{x+\delta})(\gamma-1)^2\geq 0\\
&\Leftrightarrow\gamma\leq\frac{1-p_{x+\delta}}{2-3p_{x+\delta}}.
\end{aligned}\tag{38}$$

In summarize, when the vehicle stays in the position where the scale size s satisfies $\gamma\leq\frac{1-p_{x+\delta}}{2-3p_{x+\delta}}$, the probability bound of $P(F(g(x+\delta))\neq y)$ is bounded $P(F(g(x+\delta))\neq y)\leq 1-p_{x+\delta}$.

By denoting the number of frames of the pictures as N and the size of each frame as $d\times d$, we can obtain that the picture with the same size has $2N/d$ frames. By further denoting the minimum scale size corresponding to recognizing clean pictures is $\bar{d}$, we can calculate the maximum scale times r of which the attacking probability is less than $1-p_{x+\delta}$ as:

$$\arg\max_r d\big(\frac{1-p_{x+\delta}}{2-3p_{x+\delta}}\big)^r\geq\bar{d}\tag{39}$$

to obtain $r=\lfloor\frac{\ln\bar{d}-\ln d}{\ln(1-p_{x+\delta})-\ln(2-3p_{x+\delta})}\rfloor$. Based on the times, we can then compute the number n of frames of which the attacking probability is less than $1-p_{x+\delta}$ as

$$n=\frac{2N(r+1)}{d}=\frac{2N}{d}\big(\lfloor\frac{\ln\bar{d}-\ln d}{\ln(1-p_{x+\delta})-\ln(2-3p_{x+\delta})}\rfloor+1\big).\tag{40}$$

As a consequence, by denoting the probability of successfully attacking one frame as P_a, the maximum probability of successfully attacking all frames is less than the following probability:

$$P_{\max}=P_a^N\leq(1-p_{x+\delta})^n=(1-p_{x+\delta})^{\frac{2N}{d}\big(\lfloor\frac{\ln\bar{d}-\ln d}{\ln(1-p_{x+\delta})-\ln(2-3p_{x+\delta})}\rfloor+1\big)}.\tag{41}$$