Published: 07 January 2026
Online AM: 27 October 2025
Accepted: 18 October 2025
Revised: 13 October 2025
Received: 10 August 2024

RESEARCH-ARTICLE

Invisible Adversarial Stripes on Traffic Sign: Threat and Defense for Autonomous Vehicles

DONGFANG GUO, Nanyang Technological University, Singapore City, Singapore

YUTING WU, Nanyang Technological University, Singapore City, Singapore

PENGFEI ZHOU, University of Pittsburgh, Pittsburgh, PA, United States

XIN LOU, Singapore Institute of Technology, Singapore City, Singapore

RUI TAN, Nanyang Technological University, Singapore City, Singapore

Open Access Support provided by:

Nanyang Technological University

Singapore Institute of Technology

University of Pittsburgh

Invisible Adversarial Stripes on Traffic Sign: Threat and Defense for Autonomous Vehicles

DONGFANG GUO, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore
YUTING WU, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore
PENGFEI ZHOU, Department of Informatics and Networked Systems, University of Pittsburgh, Pittsburgh, United States
XIN LOU, Infocomm Technology Cluster, Singapore Institute of Technology, Singapore, Singapore
RUI TAN, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore

Camera-based computer vision is essential to autonomous vehicle's perception. This article presents an attack that uses light-emitting diodes and exploits the camera's rolling shutter effect to create adversarial stripes in the captured images to mislead traffic sign recognition. The attack is stealthy because the stripes on the traffic sign are invisible to humans. For the attack to be threatening, the recognition results need to be stable over consecutive image frames. To achieve this, we design and implement *GhostStripe*, an attack system that controls the timing of the modulated light emission to adapt to camera operations and victim vehicle movements. Evaluated on real testbeds, GhostStripe can stably spoof the traffic sign recognition results for up to 94% of frames to a wrong class when the victim vehicle passes the road section. In reality, such attack effect may fool victim vehicles into life-threatening incidents. To counteract this threat, we propose *GhostBuster*, a software-based defense module to detect and mitigate the effects of GhostStripe. GhostBuster incorporates a perturbation detector and a sign restorer, effectively restoring the natural appearance of compromised traffic signs and significantly reducing the attack's impact. We also discuss other countermeasures at the levels of camera sensor and autonomous driving system.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Security and privacy → Systems security**; **Side-channel analysis and countermeasures;**

Additional Key Words and Phrases: Autonomous vehicle, CMOS camera sensor, rolling shutter effect, adversarial attack

A preliminary version of this work appeared in The 22nd ACM International Conference on Mobile Systems, Applications, and Services (MobiSys' 2024) [17].
This research/project is supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG4-GC-2023-006-1B).
Authors' Contact Information: Dongfang Guo (corresponding author), College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore; e-mail: dongfang.guo@ntu.edu.sg; Yuting Wu, College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore; e-mail: yuting.wu@ntu.edu.sg; Pengfei Zhou, Department of Informatics and Networked Systems, University of Pittsburgh, Pittsburgh, Pennsylvania, United States; e-mail: pengfeizhou@pitt.edu; Xin Lou, Infocomm Technology Cluster, Singapore Institute of Technology, Singapore, Singapore; e-mail: lou.xin@singaporetech.edu.sg; Rui Tan, College of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore; e-mail: tanrui@ntu.edu.sg.

ACM 1550-4859/2026/01-ART2
https://doi.org/10.1145/3773031

ACM Reference Format:
Dongfang Guo, Yuting Wu, Pengfei Zhou, Xin Lou, and Rui Tan. 2026. Invisible Adversarial Stripes on Traffic Sign: Threat and Defense for Autonomous Vehicles. *ACM Trans. Sensor Netw.* 22, 1, Article 2 (January 2026), 32 pages. https://doi.org/10.1145/3773031

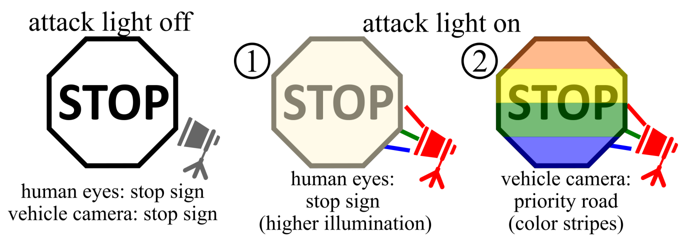
1 Introduction

Camera-based computer vision is an essential perception channel of autonomous vehicles, especially for the tasks of traffic sign recognition and lane detection [35]. Thus, reliable camera-based perception is vital to autonomous vehicle's safety. Recent research on adversarial examples [11, 15] has aroused the consciousness regarding the potential vulnerability of camera-based perception. Especially, numerous studies explore and demonstrate deployable adversarial techniques that compromise camera-based perception tasks in autonomous driving [10, 13, 22, 23, 30, 32, 37, 42, 43, 49, 60]. To better understand the security risks in this context, this article presents a physically deployable and stealthy optical adversarial-example attack that exploits the camera's rolling shutter effect to fool the car's traffic sign recognition.

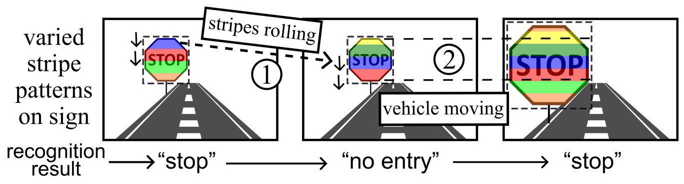
Camera sensors are based on either **charge-coupled device** (**CCD**) or **complementary metal-oxide semiconductor** (**CMOS**) technology. CCD sensors typically use global shutters, capturing an entire frame at once by exposing all pixels simultaneously. In contrast, due to CMOS's inherent line-by-line readout bottleneck, many CMOS sensors employ an electronic rolling shutter, capturing images line by line so that different pixel rows are exposed at slightly different times. While some CMOS sensors incorporate global shutters by adding additional storage elements (e.g., in-pixel memory), rolling shutter CMOS remains widely adopted in camera products, including those deployed in vehicles, because it provides a satisfactory balance between cost and image quality. For instance, both Tesla and Baidu Apollo use rolling shutter CMOS cameras in their vehicles [3, 7].

Despite its advantages, a rolling shutter CMOS camera exhibits ***rolling shutter effect*** (**RSE**) [14] when the input light contains flickering frequencies close to the operational frequency of the rolling shutter. Specifically, as the rows of a CMOS sensor are exposed in slightly different time periods, rapid changes of the input light can introduce varied color shades in different sensor scanlines and thus image distortion. Recent studies have shown the security implication of RSE, i.e., attackers can control or perturb the input light to create colored stripes on the captured image to mislead the computer vision's interpretation of the image. A recent work [44] uses **light-emitting diodes** (**LEDs**) to create flickering ambient illumination and mislead the classification of the images taken in the space under attack. In [25], a laser beamed into the camera lens creates colored stripes to disrupt object detection.

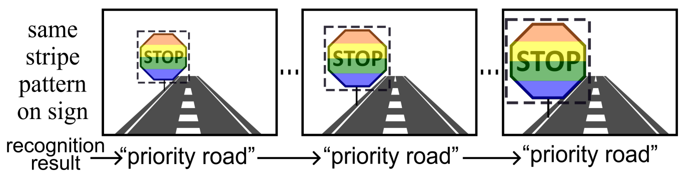
While the existing studies have implemented elementary RSE attacks on single image frames captured in controlled environments, they fall short of achieving stable attack results over a sequence of frames. This article aims to achieve stable attack results which render clearer security implications in the autonomous driving context. In the envisaged attack as illustrated in Figure 1(a), an LED is deployed in the proximity of a traffic sign plate and projects controlled flickering light onto the plate surface. As the flickering frequency is beyond the human eye's perception limit (up to 50-90 Hz [34]), the flickering is invisible to humans and the LED appears as a benign illumination device, as illustrated in Figure 1(a)-①. Meanwhile, on the image captured by the camera, as illustrated in Figure 1(a)-②, the RSE-induced colored stripes mislead the traffic sign recognition. For the attack to mislead the autonomous driving program to make erroneous decisions unconsciously, the traffic sign recognition results should be wrong and same across a sufficient number of consecutive frames. We call the attack meeting this requirement *stable*. If the attack is not stable, an anomaly detector may identify the malfunction of the recognition and activate a fail-safe mechanism, e.g., falling back to manual driving or emergency safe stopping, rendering the attack less threatening.



(a) Invisible perturbation.



(b) Unstable attack.



(c) Stable attack.

Fig. 1. Invisible optical adversarial-example attack against traffic sign recognition.

Implementing a stable attack is a non-trivial task that necessitates addressing two essential challenges, as illustrated in Figures 1(b) and (c). First, the stable attack requires the capability of stabilizing the appearance of the pre-designed colored stripes on the image cropout containing the traffic sign. Otherwise, if the stripes captured by the camera roll on the traffic sign (e.g., rolling downwards in Figure 1(b)-①), the recognition result will change over time. The rolling is caused by the discrepancy between the LED flickering frequency and the camera's rolling shutter frequency. Thus, the stripe position stabilization requires precise calibration of the LED's flickering frequency. Second, the stable attack must adapt to the time-varying position and size of the traffic sign cropout within the original image sequence captured by the moving victim vehicle. Otherwise, the stripe pattern on the traffic sign will change over time. For instance, in Figure 1(b)-②, when the stripes keep still in the **field of view** (**FoV**), the varying sign in the FoV contains varying stripe patterns, leading to varying recognition results. Thus, a stable attack, as illustrated in Figure 1(c), needs to carefully control the LED's flickering based on the information about the victim camera's operations and real-time estimation of the traffic sign position and size in the camera's FoV.

To address the aforementioned challenges in crafting a stable attack, this article presents the designs of two versions of an attack system called ***GhostStripe*** with different requirements on the attack deployment. The first version, ***GhostStripe1***, maintains stationary adversarial stripes in the FoV by calibrating the LED flickering frequency. GhostStripe1 employs a *vehicle tracker* to monitor the victim vehicle's real-time location and dynamically adjusts the LED flickering accordingly. GhostStripe1 does not require any instrumentation on the victim vehicle. It aims to maintain the victim's traffic sign recognition result stable over time. However, it is an untargeted attack, in that the recognition result is unpredictable (i.e., not deliberately spoofed towards a certain class) because the vertical positions of the adversarial stripes are not controlled by the attacker. To achieve targeted attack (i.e., the attacker can control the victim's recognition result as a certain class), on top of GhostStripe1, ***GhostStripe2*** deploys a *framing sniffer* to sense the victim camera's framing moments via a current transducer clipped on the power wire of the camera. The sniffer transmits the detected framing moments to the LED controller to refine the timing control of the flickering. Although installing the framing sniffer requires physical access to the victim vehicle, it is possible, say, during maintenance by an auto care provider colluding with the attacker.

The main contributions of this article are as follows:

— We analyze the principles for achieving stable RSE-based optical adversarial-example attack against autonomous driving perception and present techniques to satisfy the conditions obtained from the analysis.
— Following the principles, we design GhostStripe, a physically deployable attack system. Two versions of GhostStripe are designed to enable untargeted and targeted attacks with different attack deployment requirements, respectively.
— We evaluate GhostStripe on a real outdoor testbed and a lab testbed with Leopard Imaging AR023ZWDR as the victim camera, which is used in Baidu Apollo's hardware reference design [7]. On the outdoor testbed, GhostStripe1 and GhostStripe2 can achieve up to 94% and 97% success rates in launching untargeted and targeted attacks, respectively.
— We discuss various countermeasures and propose GhostBuster, a software-based defense method to detect and mitigate the threat posed by GhostStripe. With GhostBuster, we achieve a recovery in classification accuracy to 100% and 80% in over 5% and 50% of trials, respectively.

Article organization: Section 2 introduces background and preliminaries, and reviews related work. Sections 3 and 4 design and implement GhostStripe, respectively. Section 5 presents experiment setup and results. Section 6 presents our proposed defense method against GhostStripe, and discusses other possible countermeasures. Section 7 explores the attack effectiveness against traffic sign detector. Section 8 discusses several issues. Section 9 concludes this article.

2 Background and Related Work

This section provides the background and preliminaries on traffic sign recognition, rolling shutter operation, and RSE-based adversarial examples, followed by a review of related work.

2.1 Background and Preliminaries

Traffic Sign Recognition. Car-borne camera-based traffic sign recognition consists of detection and classification phases [56, 63], which are usually based on **deep neural networks** (**DNNs**). First, the detector locates the traffic sign in the image frames. Then, the detected traffic signs are cropped and fed to the classifier for interpretation. In this article, we focus on compromising the classifier.

Rolling Shutter Operation and Effect. Figure 2 illustrates the rolling shutter's operation. As CMOS sensor typically has no memory buffer to store the charge in the photodiode array, it exposes and reads out the pixel values on a row-wise basis, typically from top to bottom. Denote by N_{lines} the number of scanlines. When capturing an image frame, each scanline is exposed for a time period t_{exp}. After that, the data of the scanline is read out within a readout time denoted by t_{ro}. As illustrated in Figure 2, the exposure-readout processes for the scanlines are pipelined. The process for the next scanline is t_{ro} in time later than that of the previous scanline. As a result, the total time for capturing a frame is $t_{cap} = N_{lines} \times t_{ro} + t_{exp}$. Note that t_{ro} is fixed and can be found from the sensor specification. The t_{exp} is fixed for a certain frame but can vary across frames depending on the camera's exposure setting. The following terms are defined for the rest of this article. *Framing moment* is the time instant at which the exposure of the first scanline starts. *Frame*

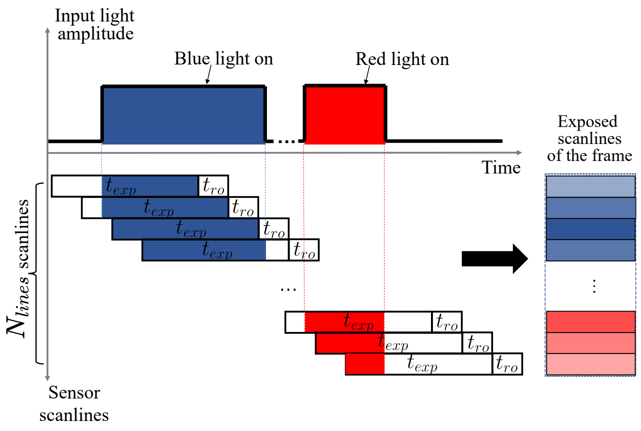


Fig. 2. Rolling shutter's operation and RSE.

period denoted by T_{frame} is the time between the framing moments of two consecutive frames, which is the reciprocal of the camera's frame rate. We have $T_{frame} \geq t_{cap}$. Now, we explain the formation of RSE. As shown in Figure 2, two light pulses affect the captured image. A pulse affects the scanlines exposed during its duration. The intensity of the affection on a scanline depends on the pulse time within the scanline's exposure. Consequently, light pulses create horizontal stripes in the captured frame.

RSE-based Adversarial Examples. An adversarial example, which is the sum of the original sample and a minute perturbation, misleads a DNN to produce a different result [15]. The work [44] presents a method that controls the LED flickering to create RSE-induced stripes as the adversarial perturbation to mislead an object recognition DNN. Its essence is as follows. Denote by $c \in \{R, G, B\}$ the color channel. We use c as the superscript of the quantity defined for a certain color channel. Denote by $t \in [0, t_{cap}]$ the relative time starting from the current frame's framing moment, by $f^c(t) \in [0, 1]$ the LED's relative emission intensity, by α^c the ambient light intensity, by β^c the LED's maximum intensity, by $l^c_{tex}(u, v)$ the texture of the scene, where (u, v) are the coordinates in the camera's FoV. Illuminated by both the ambient light and LED, the light intensity in color channel c at position (u, v) in the scene at time t is $l^c_{tex}(u, v) \cdot (\alpha^c + \beta^c f^c(t))$. From Figure 2, the exposure of the vth scanline starts at time instant vt_{ro}. Thus, the value of pixel (u, v) in color channel c is given by

$$\begin{aligned} I^c(u,v) &= \rho \int_{vt_{ro}}^{vt_{ro}+t_{exp}} l^c_{tex}(u,v)(\alpha^c + \beta^c f^c(t))\mathrm{d}t \\ &= I^c_{amb}(u,v) + I^c_{att}(u,v)g^c(v), \end{aligned}$$

where ρ is the sensor gain, $I^c_{amb}(u, v) = \rho l^c_{tex}(u, v)t_{exp}\alpha^c$, $I^c_{att}(u, v) = \rho l^c_{tex}(u, v)t_{exp}\beta^c$, $g^c(v) = \frac{1}{t_{exp}} \int_{vt_{ro}}^{vt_{ro}+t_{exp}} f^c(t)\mathrm{d}t$. Note that $I^c_{amb}(u, v)$ is the image in color channel c captured with ambient illumination only. The $I^c_{att}(u, v)$ is the image captured with light emitted from the LED in full intensity all the time and no ambient illumination. It can be obtained by $I^c_{att}(u, v) = I^c_{full}(u, v) - I^c_{amb}(u, v)$, where $I^c_{full}(u, v)$ is the image captured with both the ambient illumination and the full-intensity light from the LED. Both $I^c_{amb}(u, v)$ and $I^c_{full}(u, v)$ are collected by the attacker in advance. The LED control signal in all color channels $f(t) = \{f^R(t), f^G(t), f^B(t)\}$ is designed by solving $\operatorname{argmin}_{f(t)} \ell(\mathcal{M}(I(u, v)), k)$, where $I(u, v) = \{I^R(u, v), I^G(u, v), I^B(u, v)\}$, $\mathcal{M}(\cdot)$ is the classifier, k is the target class of the attack (i.e., the attack aims to mislead the classifier to produce class k), $\ell(\mathcal{M}(I(u,v)),k)$ is the classification loss for the target class k when the classifier is fed with $I(u,v)$.

2.2 Related Work

Physical attacks on general computer vision. There are a number of work that studies the practical physical adversarial attacks against computer vision tasks [50, 51]. The physical forms of the attacks include but are not limited to stickers/patches/clothes [9, 13, 52], 3D-printed objects [8], and glasses [45]. Unlike prior digital adversarial attacks, physical attacks are usually visible in the real-world to both human eyes and cameras, as they need to be effectively-captured by the camera (although might be designed to be less noticeable/unreasonable). Different from conventional physical attacks that aim to compromise general computer vision tasks, GhostStripe is specifically designed for autonomous vehicle's camera perception, and the perturbation is invisible to human eyes.

Physical attacks on autonomous vehicle camera perception. Physical attacks can be categorized into *object perturbation* and *camera perturbation*. Object perturbation attacks modify the object appearance, such as using stickers or light on traffic signs to mislead recognition [13, 30], painting on billboards to alter steering [60], 3D-printed objects to escape detection [10], patches/marks on roads to confuse lane detection [23, 43], and depth-less images recognized as real objects [37]. These attacks are visible to humans. Camera perturbation attacks exploit camera hardware properties, like using lasers to blind cameras [39, 53], projecting patterns to create lens flare effects [32], and using infrared light to create magenta pixels [49]. These require precise targeting of the camera lens. In contrast, GhostStripe uses the traffic sign to reflect attack light, requiring no such physical maneuvers. A recent work [42] uses invisible infrared lasers to reflect projections on traffic signs, creating perturbations as purple/magenta spots, effective only for cameras without infrared filters. Another approach [22] uses sound waves to interfere with image stabilizer's built-in inertial sensor, triggering unwanted motion compensation, but it only disrupts object detection in single frames and does not address attack stability.

RSE applications and exploitation for attacks. Many **visible light communication (VLC)** systems are designed based on RSE [12, 18, 19, 27, 55, 57]. These systems encode information through controlled flickering of light sources, which cameras decode from the induced stripes. Such a VLC capability can be employed in indoor smartphone localization using LED landmarks [26, 40]. RSE has also been applied for watermarking physical scenes to prevent unauthorized photographing [58, 62], and for enhancing hand pose tracking and reconstruction [59].

In addition to [44] that is employed as a baseline attack method in this article, a few other works [25, 28, 54] also exploit RSE to mislead computer vision. The work [28] explores the RSE-based backdoor attack. During training data collection, light flickering creates RSE-induced stripes to embed poisoning samples with adversarial class labels. During inference, the same flickering triggers the backdoored classifier to output the adversarial class. The works [25, 54] particularly consider RSE-based attacks in the context of autonomous vehicles. The work [25] models the rolling shutter process by collecting RSE patterns with various parameter settings in a dark room. Certain RSE patterns overlaid on captured images can lead to miss detection of up to 75% objects. In an autonomous vehicle simulator, the attack can introduce noticeable braking delays when a pedestrian or cyclist is in front of the vehicle under attack. The work [54] uses a laser to cause a monochromatic stripe that covers the traffic light to disturb the traffic light color recognition. The laser's emission duration is controlled based on the frame time. However, these two attacks [25, 54] require aiming the laser at the victim vehicle's camera lens, while GhostStripe is free of this requirement. Moreover, the above works [25, 28, 54] do not consider the phase synchronization issue discussed in Section 3.1. Thus, they cannot control the positions of the RSE-induced stripes. Differently, GhostStripe2 applies framing sniffer to achieve phase synchronization.

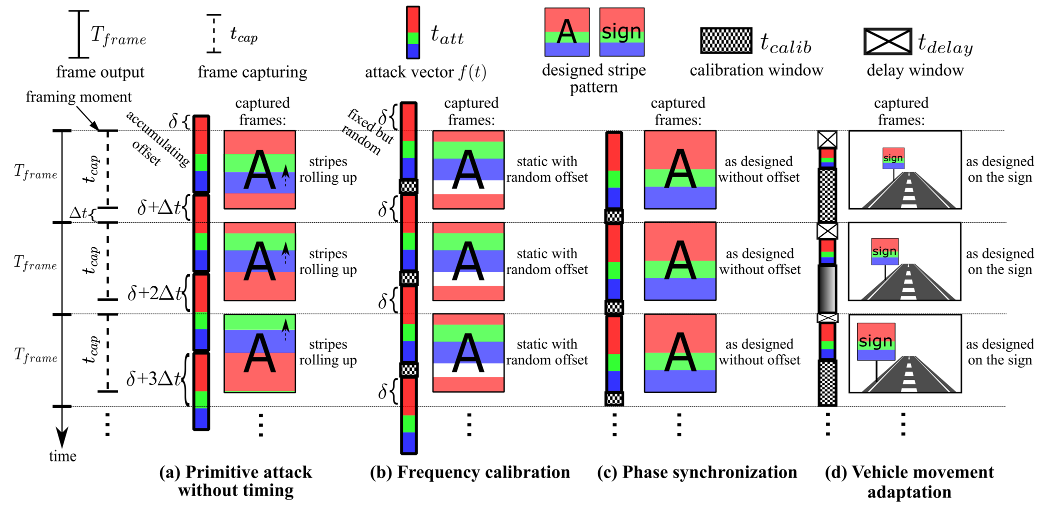

T_{frame} frame output
t_{cap} frame capturing
t_{att} attack vector $f(t)$
designed stripe pattern
t_{calib} calibration window
t_{delay} delay window
framing moment
accumulating offset
fixed but random
captured frames:
stripes rolling up
static with random offset
as designed without offset
as designed on the sign
δ
$\delta+\Delta t$
$\delta+2\Delta t$
$\delta+3\Delta t$
Δt
time
(a) Primitive attack without timing
(b) Frequency calibration
(c) Phase synchronization
(d) Vehicle movement adaptation

Fig. 3. Illustrations of the designs of attack timing control and vehicle movement adaptation.

Our prior work [17] introduces the design of GhostStripe and provides an evaluation of its attack performance. Building on that foundation, this article makes the following contributions: (1) We design and evaluate GhostBuster, a defense module to detect and mitigate the threat posed by GhostStripe. (2) We extend our evaluation on GhostStripe by investigating the target class feasibility and the opportunities to bypass an entropy-based anomaly detector. (3) We explore GhostStripe's attack effectiveness against traffic sign detector. (4) We include a broader review of related work.

3 Design Principles of GhostStripe

This section analyzes two principles to achieve stable attack described in the introduction section, i.e., *attack timing control* and *vehicle movement adaptation*.

3.1 Attack Timing Control

In this section, we analyze the simplified scenario described in Section 2.1, i.e., the whole images in a frame sequence are classified. Figure 3(a)–(c) depicts our analysis in this section. In reality, the vehicle classifies a sequence of image cropouts containing the traffic sign, as illustrated in Figure 3(d). In Section 3.2, we will analyze how to deal with this real scenario.

To affect consecutive frames, the attacker needs to keep replaying the designed attack signal $f(t)$ where $t \in [0, t_{cap}]$ to control the LED. Note that $T_{frame} \geq t_{cap}$ and we define $\Delta t \triangleq T_{frame} - t_{cap}$. In addition, we use δ to denote the time offset between the onset moment of the first play of $f(t)$ and the nearest camera's framing moment. A primitive attack, which continuously replays $f(t)$ back to back, accumulates Δt over time on the offset between the replay's onset moment and the camera's framing moment. As illustrated in Figure 3(a), the offset increases by Δt for every frame. The resulting stripe pattern created by the attack rolls across the FoV over time (e.g., roll up in Figure 3(a)), leading to varying classification results.

To achieve a stable attack, the rolling needs to be avoided by *frequency calibration* such that the replay frequency is identical to the frame rate. This can be achieved by adding a calibration period $t_{calib} \triangleq T_{frame} - t_{cap}$ after each replay, as illustrated by the checkerboard squares in Figure 3(b). As such, the offset between the replay's onset moment and the camera's framing moment is fixed at δ over frames. The δ can take any value from $[-T_{frame}/2, T_{frame}/2]$, depending on the onset time of the attack. The resulted stripe pattern is stationary in the FoV, but the position offset is uncertain. This uncertainty renders the attack untargeted.

If the attacker can further control its attack onset time such that $\delta = 0$ (which is called *phase synchronization*), the RSE-induced stripes will be identical to the designed pattern, as illustrated in Figure 3(c). Hence, the victim's classification results over frames will be the target class k. To perform the phase synchronization, the attacker needs to obtain the framing moments, which can be sensed from the victim camera's magnetic emanation, as we will detail in Section 4.5.

3.2 Vehicle Movement Adaptation

The traffic sign recognition pipeline only classifies the image cropout containing the detected sign. Thus, only the RSE-induced stripes within the cropout affect the classification. As the cropout's position and size in the FoV vary when the vehicle moves, the attack needs to adapt to the vehicle's movement. The adaptation logistics is analyzed as follows.

Assume the cropout's upper edge is at the N_{up}th scanline and its vertical dimension is N_{sign} scanlines. For ease of explanation, we analyze the case with phase synchronization. As illustrated in Figure 3d, the attack uses three time windows: *delay window*, *attack window*, and *calibration window*, represented by crossed, colored, and checkerboard squares, respectively. Their lengths are $t_{delay} = (N_{up} - 1) \times t_{ro}$, $t_{att} = N_{sign} \times t_{ro} + t_{exp}$, and $t_{calib} = T_{frame} - t_{delay} - t_{att}$. The malicious LED flicking is performed within t_{att}. When the victim vehicle moves, the t_{delay}, t_{att}, and t_{calib} change over frames, maintaining the stripe pattern as designed on the sign cropout over frames. For each frame, the LED control signal $f(t)$ during t_{att} can be designed by solving $\operatorname{argmin}_{f(t)} \ell(\mathcal{M}(I_{cropout}), k)$, where $I_{cropout}$ is the RSE-affected image cropout. However, high compute overhead makes online solving impractical. To simplify, we design an LED control signal $f_0(t)$ for a minimum attack window t_{att0} during the offline stage, based on the smallest detectable traffic sign in the FoV. At runtime, when $t_{att} \geq t_{att0}$, $f(t)$ is scaled up by t_{att}/t_{att0} and replayed. The replayed attack light signals can be filled into the calibration and delay windows to ensure that the perturbations appear on the traffic sign when there is no phase synchronization, and avoid noticeable on-off flickering at the frame rate.

4 GhostStripe Design

This section presents the design of GhostStripe. We first summarize the basic attack assumptions in Section 4.1. Then, we overview the two versions of GhostStripe in Section 4.2. Then, the remaining three subsections present the approaches to attack signal optimization, vehicle movement adaptation, and phase synchronization, respectively.

4.1 Basic Attack Assumptions

The attacker assumptions are: (1) The attacker can deploy a malicious LED to illuminate the traffic sign and a *vehicle tracker* to monitor the road section. (2) The attacker knows fixed parameters of the victim vehicle's camera: focal length, sensor size, image resolution, and frame rate, which are obtainable from datasheets and reverse engineering [22, 25, 32, 49, 54]. For cameras with auto-exposure, the attacker can model the relationship between exposure time and ambient light [25]. (3) The attacker has either white-box or black-box access to the DNN used for traffic sign recognition. White-box access includes knowledge of the DNN's architecture and weights, while black-box access involves only the executable DNN without internal information. Obtaining DNNs might be harder but is assumed in all white-box [10, 13, 23, 30, 32, 60] and black-box [22, 25, 30, 32, 54] attacks. It is possibly achievable through open codebases, reverse engineering, or social engineering.

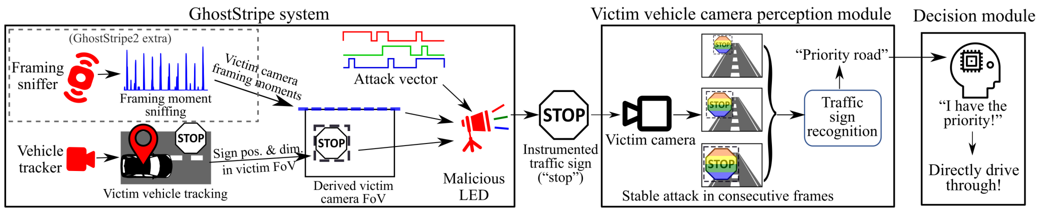


Fig. 4. Overview of GhostStripe.

4.2 System Overview

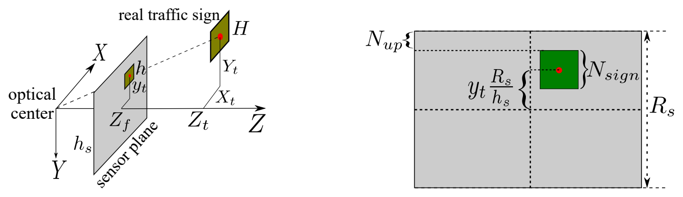
We design two versions of GhostStripe, i.e., GhostStripe1 and GhostStripe2, with different requirements on the attack deployment to achieve untargeted and targeted stable attacks, respectively. GhostStripe1 maintains stationary adversarial stripes within the victim FoV by calibrating the LED flickering frequency and performs vehicle movement adaptation for real-time adjustment. It achieves an untargeted attack. On top of GhostStripe1, GhostStripe2 implements phase synchronization to eliminate the random offset δ. Therefore, the resulting adversarial stripe pattern remains same as designed and misleads the victim to produce the target class k. To achieve the phase synchronization, GhostStripe2 requires clamping a sensor called *framing sniffer* onto the victim vehicle's camera power wire to sense the framing moments. Therefore, it targets a specific victim vehicle and controls the victim's traffic sign recognition results.

During the offline attack preparation phase, the attacker designs an LED control signal $f_0(t)$ for a minimum attack window t_{att0} as described in Section 3.2. The workflow of GhostStripe during the online attack execution phase is illustrated in Figure 4. The vehicle tracker tracks the victim vehicle's real-time position and estimates the traffic sign's position and dimension in the victim vehicle camera's FoV. In GhostStripe2, the framing sniffer senses the framing moments from the magnetic emanation of the camera power wire. Both the vehicle tracker and framing sniffer continuously transmit their sensing results to the LED controller. Whenever the LED controller receives a report from either the vehicle tracker or the framing sniffer, it updates the attack signal and control parameters. Specifically, it scales up $f_0(t)$ to have $f(t)$ according to the traffic sign's dimension and also determines the three time windows for attack timing control as illustrated in Figure 3(d) and Section 3.2. The LED controller continuously replays the latest $f(t)$ with attack timing control.

4.3 Attack Signal Optimization

This section describes the generation of the minimum LED control signal $f_0(t)$. To improve the robustness of the attack, $f_0(t)$ is obtained by solving $\operatorname{argmin}_{f_0(t)} \mathbb{E}_\phi[\ell(\mathcal{M}(I^\phi_{sign}), k)]$, where ϕ represents the uncontrollable offset in terms of the number of scanlines; $I^\phi_{sign}(u,v) = I_{sign,amb}(u,v) + I_{sign,att}(u,v) \cdot g(v+\phi)$ is the image cropout containing the traffic sign; $I_{sign,amb}(u,v)$ and $I_{sign,att}(u,v)$ are the corresponding image cropouts from $I_{amb}(u,v)$ and $I_{att}(u,v)$ defined in Section 2.1. For GhostStripe1, since there is no control on the offset, we sample ϕ uniformly from $[0, N_{sign}]$ to evaluate the mathematical expectation of the objective function; for GhostStripe2, as the phase synchronization can largely reduce the offset, we sample ϕ uniformly from a narrow range of $[-0.1N_{sign}, 0.1N_{sign}]$, where the multiplier 0.1 is empirically chosen.

White-box optimization. Since the analytical model of the rolling shutter as described in Section 2.1 is differentiable, $f_0(t)$ can be obtained by gradient-based methods. We use **Projected Gradient Descent (PGD)** [31], which iteratively perturbs input data towards maximizing the loss function while maintaining the perturbations within a bounded range, i.e., $f_0(t) \in [0,1]$.



(a) Prospective projection model.

(b) Traffic sign's projection in the sensor plane.

Fig. 5. Estimation of the traffic sign's vertical position and size in the captured image.

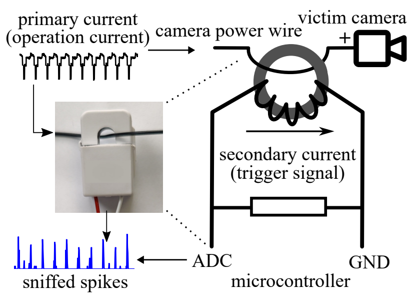
By iteratively adjusting the $f_0(t)$ based on the attainable internal gradients, PGD can efficiently optimize the $f_0(t)$ against the victim model.

Black-box optimization. We implement **Bayesian Optimization** (**BO**) [36, 38], which is a strategy for global optimization of black-box functions. It involves a Bayesian statistical model and an acquisition function. The statistical model generates a Bayesian posterior probability distribution to approximate the objective function, updated with each new query. Subsequently, this posterior distribution is utilized to construct the acquisition function, determining the next query point. With black-box access, we query the model with attacked images $I(u, v)$, and obtain prediction classes and confidence outputs. This allows BO to iteratively refine $f_0(t)$ based on the model's responses. Since BO is suitable for problems in low cardinality (typically, lower than 30), we reduce the cardinality of $f_0(t)$ by restructuring each color channel $f_0^c(t)$ as a vector of length q. Each element lasts for a time period t_{att0}/q. This limits BO's search space dimension to $3 \times q$ for the three color channels of $f(t)$. In terms of perturbation appearance, the final perturbation consists of q stripes with equal vertical length, in contrast to the stripes in the white-box setting that are on a scanline-wise basis. In our implementation, we experimentally choose q from 5 to 10 and use the one that yields the best attack effectiveness.

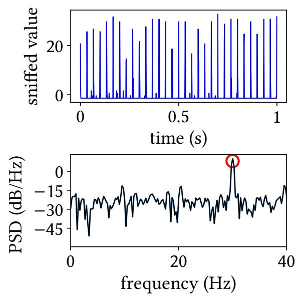
4.4 Locating Traffic Sign in Camera FoV

This section presents the approach to estimating the traffic sign's vertical position and size in the victim vehicle camera's FoV based on the prospective projection model. Figure 5(a) shows an *ego coordinate system* originating from the victim camera's optical center. The X- and Y-axes define the camera sensor plane, and the Z-axis is the optical axis perpendicular to it. Let (X_t, Y_t, Z_t) and H denote the coordinates of the traffic sign's center and the vertical dimension of the traffic sign, respectively. Z_f and h_s denote the victim camera's focal length and the vertical sensor dimension. From Figure 5(a), the traffic sign's vertical position and size on the sensor plane are $y_t = Z_f \frac{Y_t}{Z_t}$ and $h = Z_f \frac{H}{Z_t}$. With R_s as the total number of camera scanlines, a unit length of the sensor plane's vertical dimension corresponds to $\frac{R_s}{h_s}$ scanlines. Figure 5(b) shows the sensor plane and the traffic sign's projection. The projection's vertical size and position in scanlines are $N_{sign} = h\frac{R_s}{h_s} = Z_f \frac{H}{Z_t}\frac{R_s}{h_s}$ and $N_{up} = \frac{1}{2}R_s - y_t\frac{R_s}{h_s} - \frac{1}{2}N_{sign} = \frac{1}{2}R_s - (Y_t + \frac{1}{2}H)\frac{Z_f R_s}{Z_t h_s}$. The values of Z_f, h_s, and R_s are available from the camera's datasheet; the traffic sign size H can be measured by the attacker.

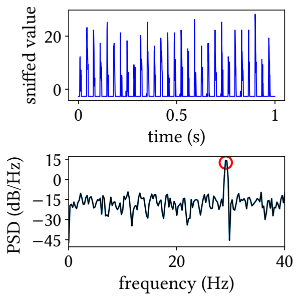
From above, to estimate N_{sign} and N_{up}, the attacker needs to obtain Y_t and Z_t. If the road section is flat, Y_t is the altitude difference between the traffic sign and the vehicle camera. The traffic sign's altitude can be measured by the attacker; the vehicle camera's altitude can be obtained from the vehicle specification or measured by the attacker. Z_t is the horizontal distance between the victim vehicle and the traffic sign, which can be obtained by localizing the victim vehicle in real time. With



(a) Framing sniffer.



(b) Sniffed trace and PSD of LI-AR023ZWDR.



(c) Sniffed trace and PSD of Arducam AR1820HS.

Fig. 6. Framing sniffer and measurement traces.

Z_t, the updated N_{sign} and N_{up} are used for vehicle movement adaptation. The victim camera's pitch angle and road gradient can be obtained from specification or measurement, and can be factored in when determining Y_t.

4.5 Phase Synchronization

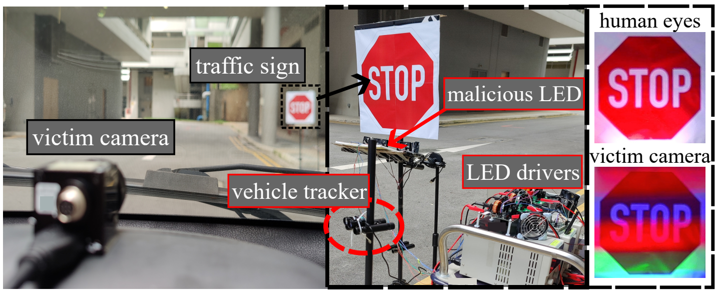
This section presents how GhostStripe2 senses the victim camera's framing moments to achieve phase synchronization. The internal operations of a camera may create variations in the camera's current draw and the resulting magnetic emanation. We investigate whether the emanation provides salient characteristics for inferring framing moments. To sense the magnetic emanation, as shown in Figure 6(a), we integrate a YHDC SCT-006 split-core current transducer with a 330 Ω resistor and sample the voltage over the resistor using an Arduino Due. The current transducer is clamped onto the camera's power wire. The current in the wire generates a magnetic field concentrated at the magnetic split-core, which further induces a secondary current in the winding and then a voltage over the resistor. Figure 6(b) shows the measurement trace for the Leopard Imaging AR023ZWDR camera, which is the camera product in Baidu Apollo's hardware reference design [7]. We can see periodic time-domain spikes with intervals about T_{frame}. Figure 6(b) also shows the **power spectral densities** (**PSDs**) of the measurement trace. The highest PSD peak appears at the camera's frame rate. Figure 6(c) presents the measurement trace and PSD for another camera (Arducam AR1820HS), revealing a similar pattern. These results suggest that the time-domain spikes may be indicative of framing moments. The sniffer uses a threshold to detect the time-domain spikes. Upon detecting a spike, the sniffer transmits a packet to the LED controller via two Nordic nRF24L01+ transceivers operating in the 2.4 GHz ISM band, which then prompts the replay of the light signals upon packet detection. By cross-validation across two AR023ZWDR cameras, we show that the sniffed time-domain spikes can be used to synchronize the LED with the camera's framing moment with proper profiling. Details including the synchronization validation can be found in our prior work [17].

5 Evaluation

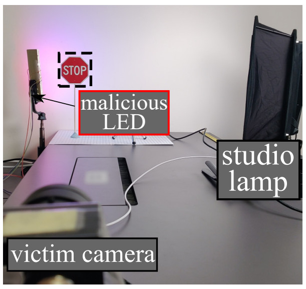
We evaluate GhostStripe's attack effectiveness by testing it against the camera on a moving vehicle in the outdoor testbed. Throughout this section, we use the abbreviation ***GS*** to refer to GhostStripe.

5.1 Testbed Setups

Outdoor testbed: We use Leopard Imaging AR023ZWDR as the victim camera, which is the default main camera in Baidu Apollo's hardware reference design [7]. It is built upon the ONSEMI AR023Z



(a) Outdoor testbed.



(b) Lab testbed.

Fig. 7. Testbed setups.

rolling shutter image sensor with a size of 5.78 mm × 3.26 mm, 1928 × 1088 active pixels. Each scanline has a readout time t_{ro} of 30 μs. Its focal length is 12 mm. We use a real road section and a real car, as shown in Figure 7(a). We deploy most common traffic signs [2] including "stop", "yield", and "speed limit" with size and altitude conforming to the **Manual on Uniform Traffic Control Devices** (**MUCTDs**) [6]. To evaluate the impact of traffic sign textures, we also include a "stop" sign coated with a highly reflective sheeting. We mount the victim camera under the front windshield of the car. The sign-car distance for the camera to perceive the whole sign is from 10 m to 32 m.

Lab testbed: We also build a lab testbed in 1:10 scale as shown in Figure 7(b) to isolate the impact of uncontrollable environment factors and provide better understanding of the impacts of several factors on *GS*. The total length of the testbed is 3.6 m. To control ambient illumination condition, we set up two studio lamps with tunable intensity to project light onto the testbed. The color temperature of the lamps is 5, 600 K, which is similar to normal sunlight. In addition to the four common signs described above, which use red as their primary color, we include two additional signs with more diverse colors, i.e., "priority road" (yellow) and "ahead only" (blue). In addition to the AR023ZWDR camera, we also include an Arducam AR1820HS for lab testing. The AR1820HS is built on the ONSEMI AR1820HS rolling shutter sensor, which measures 6.14 mm × 4.60 mm and contains 4912 × 3864 active pixels. We test it under the 1920 × 1080 resolution mode, which supports a typical frame rate of 29 FPS. The scanline readout time is $t_{ro} = 27$ μs. We equip this camera with lenses of different focal lengths (3.9 mm, 6 mm, 12.5 mm, and 25 mm).

Traffic sign recognition models. We integrate the YOLO object detector [41] and an AlexNet-based 8-layer convolutional neural network traffic sign classifier. We train the classifier on the **German Traffic Sign Recognition Benchmark** (**GTSRB**) dataset [46]. The trained model achieves a 95.35% accuracy on the GTSRB testing set. When tested on video frames taken for the signs deployed in our testbeds without attack, it achieves 100% accuracy across various camera poses, distances, and illumination conditions considered in our experiments. To evaluate the attack generality and transferability across different victim models, we also employ a ResNet-based classifier, which achieves 97.24% accuracy on the GTSRB testing set and 100% accuracy on our collected attack-free data.

GhostStripe Implementation. We implement GhostStripe by following the workflow presented in Section 4.2. The replay of a given $f(t)$ is implemented by **pulse-width modulation** (**PWM**) for the LED's power supply using an Arduino Due. We integrate 30 Marktech XM-L RGB LED units to emit the attack light. We implement an essential victim vehicle localization function based on a LightWare SF30/C LiDAR rangefinder placed on the road side. More details on the LED driver and the vehicle tracker can be found in our prior work [17].

5.2 Evaluation Metrics and Baselines

We use the following metrics to characterize attack effectiveness: (1) **Misclassification rate (MR)**: The ratio of frames where the traffic sign is incorrectly identified, divided by the total number of frames. (2) **Primary misclassification class rate (PMCR)**: The primary misclassification class is defined as the most frequently misclassified class when *GS1* is deployed, or the targeted class when *GS2* is deployed. PMCR is the ratio of frames where the traffic sign is misclassified as the primary misclassification class to the total number of frames. (3) **Entropy**: We employ Shannon entropy to quantify the randomness of classification results within a time window. In this section, we compute the entropy values within 1.5 s time windows, adopted from the window size for decision making used in Baidu Apollo's traffic light recognition. Lower entropy values signify increased stability in classification results.

We employ the following baseline attack approaches: (1) ***Random***: Employs randomly appearing colored stripes. (2) ***Primitive*** [44]: Generates the colored stripes with an offset-robust design which is also used in *GS1* as described in Section 4.3, without timing control for stable attack. (3) ***GS2-still***: A variant of *GS2* designed for a specific victim location without vehicle movement adaptation, used to assess the impact of vehicle movement adaptation on attack performance.

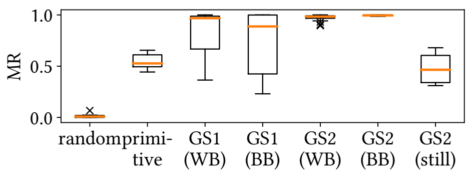
5.3 Evaluation on Outdoor Testbed

5.3.1 Impact on Detection. We assess *GS*'s impact on traffic sign detection (i.e., the step prior to classification). We measure the **Intersection over Union** (**IoU**) between detection results with and without attack at different vehicle-sign distances, as shown in Figure 9. The detector achieves a consistently high IoU of around 0.94 during the *GS* attack. When using these detection results to select cropouts from clean images when the attack is temporarily switched off, all cropouts are correctly classified. Thus, *GS* has negligible impact on the traffic sign detector.

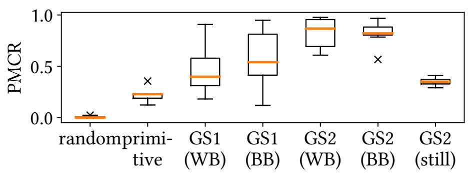
5.3.2 Overall Attack Performance. We evaluate GS against a moving vehicle using the most representative sign "stop" as an example. In this subsection, the attack is based on $t_{exp} = 1/1,000$ s. During the offline attack optimization phase, *Random* rarely deviates classification results from the ground truth. With *Primitive* and *GS1* which share the same attack signals optimized for the entire offset range, the untargeted attack succeeds at 87.2% in the white-box setting and 81.1% in the black-box setting. For *GS2*, we choose the "priority road" sign as the target class. *GS2* achieves a 100% targeted **attack success rate** (**ASR**) in both white-box and black-box settings.

Then, we test the attacks on the testbed during normal daytime hours (9 am to 5 pm) under partly cloudy weather conditions. The vehicle drives along the road at around 10km/h, recording video footage containing the traffic sign under attack. Figure 8 summarizes the overall attack performances. *Random* is ineffective, with both MR and PMCR nearly zero. *Primitive* achieves a mean MR of 54.5% and PMCR of 22.4%, but with a high mean entropy of 2.55, indicating unstable classification results within each 1.5, s window due to varied stripe patterns on the sign cropout across frames.

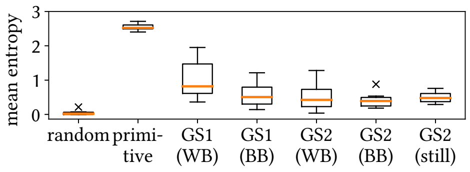
Both *GS1* and *GS2* perform effectively, regardless of whether they are generated with white-box or black-box (indicated as "WB" and "BB" in Figure 8, respectively) DNN knowledge. *GS2* exhibits the highest performance in targeted attacks, achieving mean PMCRs of 83.2% under the white-box setting, and 82.4% under the black-box setting. Here the PMCRs of white-box setting show more variation than black-box setting. This is likely due to the varying testing conditions across trials. While the white-box attack requires more information, its main benefit lies in optimization efficiency. After successful training, white-box attack is not necessarily more effective than black-box at runtime, as effectiveness depends on testing conditions. *GS1* demonstrates a high success rate in untargeted attacks, with mean and median MRs of 81.5% and 96.8% under the white-box



(a) MR.



(b) PMCR.



(c) Mean entropy.

Fig. 8. Comparison with baseline methods. Abbreviations: "WB" for "white-box", and "BB" for "black-box".

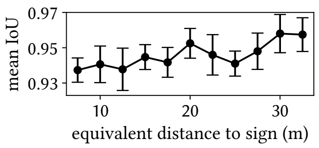


Fig. 9. Impact on front-end detection.

setting and 73.4% and 88.7% under the black-box setting. Note that the primary misclassification class in *GS1* may vary across trials as different perturbation offsets may result in different classes. Although the PMCRs of *GS1* hover at around 50%, which are lower than *GS2*, they are still higher than other methods. The relatively low PMCR of *GS1* compared with *GS2* is explained as follows. During the *GS1*'s offline attack signal optimization, the vertical offset ϕ is sampled from a wide range. As such, adjacent offsets may not result in the same class. Consequently, at runtime, when slight misalignments occur between the designed stripes and the sign cropout in the victim FoV, the misclassification results may vary. However, the relatively stable stripe pattern in *GS1* still contributes to overall attack stability, as indicated by the slight entropy increase compared with *GS2*.

We also compare *GS1* and *GS2* in Figure 10. For *GS2*, the minimum MR and PMCR, and maximum mean entropy are 89.5%, 56.6%, and 1.28. On *GS1*'s **cumulative distribution function (CDF)** curves, the corresponding probabilities are 49%, 65%, and 83%, as illustrated in Figure 10(b) and (c). The interpretation of these results are as follows. In terms of MR, *GS1* performs no worse than *GS2* in 100% − 49% = 51% cases for spoofing traffic sign to any other class during one run. In terms of PMCR, *GS1* performs no worse than *GS2* in 100% − 65% = 35% cases for spoofing traffic sign to a primary misclassification class during one run, although this class is not controllable. In terms of entropy within each time window, *GS1* performs no worse than *GS2* in 83% cases.

GS2-still achieves 48.2% mean MR, 35.3% PMCR, and 0.50 mean entropy. The performance drop compared with *GS2* is because when the stripes fall on the traffic sign in the FoV, the attack is targeted; otherwise, the results are unpredictable. This shows the benefit of continuous vehicle tracking and movement adaptation, for enhancing attack effectiveness compared with a static attack targeting a specific position.

5.3.3 Visualization of Attack Effectiveness. We illustrate the attack effectiveness of the attack results by drawing the classification results when the vehicle drives through the road section, as

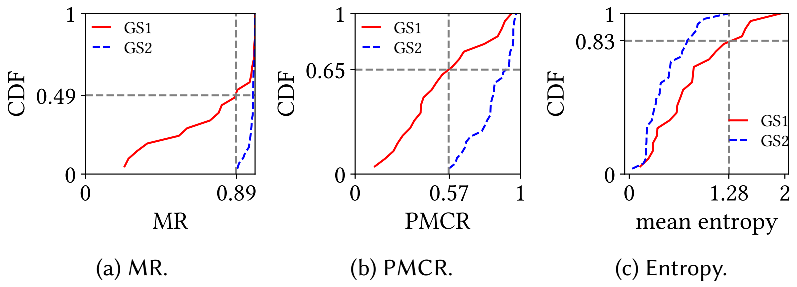


Fig. 10. Comparison between GS1 and GS2.

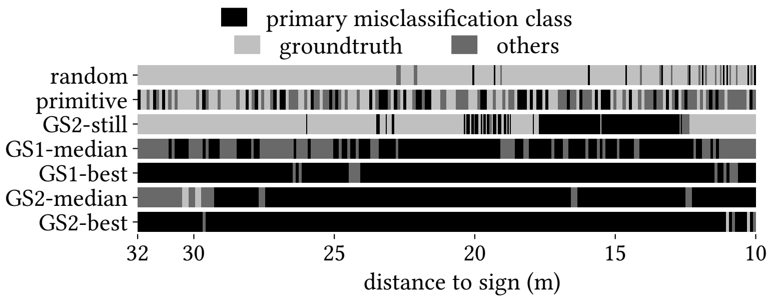


Fig. 11. Examples of attack results on the consecutive frames when the vehicle passes the road section.

shown in Figure 11. *GS1-median* and *GS2-median* denote the result traces in the runs where *GS1*'s and *GS2*'s PMCRs are around their respective median levels. *GS1-best* and *GS2-best* denote the best result traces of *GS1* and *GS2* in all runs. Both *GS1* and *GS2* achieve relatively stable attack effectiveness. In the best cases, *GS1* and *GS2* can achieve ASRs of over 94% and 97%, respectively, in misleading the victim to the primary misclassification class stably. In contrast, baseline attack methods show ineffectiveness and/or result randomness.

5.3.4 *Impact of Distance.* We use *GS2* to understand the impact of the distance between the sign and moving vehicle on the attack effectiveness. We divide the road section into 22 one-meter segments and calculating the metrics for each. Figure 12 shows the results. When the camera first perceives the sign, MR reaches 77.6% while PMCR is 46.7%. As the vehicle approaches the sign, both MR and PMCR increase, exceeding 97% and 80% within 25 meters. The decrease in attack effectiveness at greater distances is likely due to reduced attack light intensity. Additionally, at greater distances, the smaller N_{sign} makes the stripes on the sign more vague due to the reduced time difference between the exposure of adjacent vertical portions, causing the light signal to affect these portions more similarly. The performance degradation may be mitigated by increasing attack light intensity. Besides, perception results closer to the sign may be more critical for driving decisions, as newer perception results may override earlier ones.

5.3.5 *Impact of Movement Speed.* We use *GS2* to study the impact of vehicle movement speed on the attack effectiveness. We test with speeds at around 10, 20, and 30 km/h, separately. Figure 13 shows the mean PMCR and entropy versus vehicle speed. We do not observe a noticeable relationship between the attack performance and speed.

5.3.6 *Sign Classes and White/Black-box Attack.* We evaluate the feasibility of *GS* against different groundtruth and targeted classes in a stationary setting at a sign-camera distance of 16 m. We select

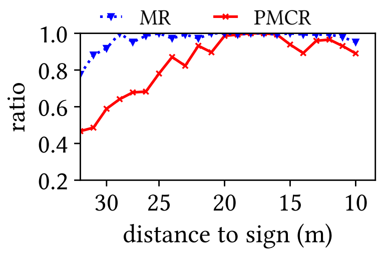


Fig. 12. Impact of sign-vehicle distance.

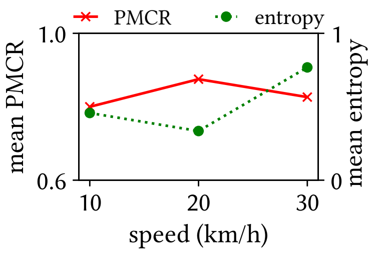


Fig. 13. Impact of victim vehicle's speed.

Table 1. *GS1*'s Effectiveness on Most Common Traffic Signs

Original	**MR**	
	White-box	**Black-box**
Stop	89.8%	81.9%
Yield	54.8%	0%
Speed limit 30 km/h	92.3%	73.3%
Speed limit 80 km/h	75.0%	70.8%

the most common signs, including "stop", "yield", and "speed limit" [2]. Table 1 lists the overall attack effectiveness of *GS1* on these traffic signs. Table 2 lists the target classes that are semantically conflicting with white-box PMCRs over 60% in *GS2*. The target classes for *GS2* are not arbitrary for each original sign due to the constraints of the perturbations' stripy forms. The results show that it is possible for the attacker to design specific attack scenarios (e.g., speed-up attack, sudden-braking attack, sign-ignoring attack) against the victim according to the expected attack consequence.

We also examine the relationship between inter-class similarity and target class feasibility. Table 2 lists and ranks the target classes confidences achieved from the victim model's output when the original sign is attack-free and correctly classified. The attack-free confidences of each target class can then be a representation of the inter-class similarity in the latent space extracted by the victim model. The higher the attack-free confidence, the more similar the target class is towards the original class. Table 2 ranks the target class confidences achieved from the victim model's output when the original sign is attack-free and correctly classified. These attack-free confidences represent inter-class similarity in the latent space extracted by the victim model. Higher attack-free confidence indicates greater similarity to the original class. Results show that the inter-class similarity may be related to target class feasibility. For example, for all four original signs, the most similar classes (ranked 2nd) are feasible targets. It is harder to compromise the "yield" sign, likely due to its distinct inverted triangle shape different from others, as indicated by the very low confidence across other classes (e.g., the ranked 2nd confidence is already as low as 4.7×10^{-17}). However, achieving high confidence or rank is not necessary for a feasible target class. This is because the stripy perturbation significantly disturbs the original image, introducing substantial variation in the features in the embedded space. The attacker can determine the feasible set of target signs by training for each semantically-conflicting sign and select the applicable ones according to the expected attack scenarios.

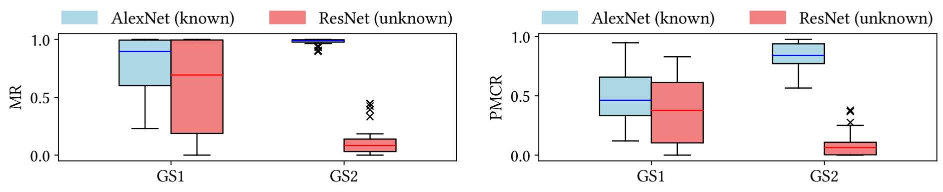
Table 2 also compares the attack effectiveness obtained under the white-box and black-box settings. The training of a black-box attack is more challenging to converge to some targeted classes than white-box attack. This is because the black-box attack faces more constraints such as stripe

Table 2. *GS2*'s Effectiveness on Most Common Traffic Signs

Original	Target	PMCR		Attack-free confidence	
		White-box	Black-box	Rank	Confidence
Stop	Speed limit 80 km/h	100%	99.2%	2	6.9×10^{-3}
	Speed limit 30 km/h	99.1%	99.9%	4	2.1×10^{-5}
	Speed limit 20 km/h	70.6%	71.7%	6	9.4×10^{-6}
	Right-of-way	96.6%	92.8%	8	4.6×10^{-6}
	Priority road	99.3%	98.9%	27	3.7×10^{-10}
	End of no passing	72.8%	22.4%	29	1.3×10^{-10}
Yield	Priority road	97.3%	10.1%	2	4.7×10^{-17}
Speed limit 30km/h	Keep right	98.1%	99.6%	2	1.2×10^{-5}
	End of speed limit 80 km/h	97.2%	97.5%	3	8.3×10^{-7}
	Speed limit 80 km/h	100%	99.9%	4	2.2×10^{-7}
	Speed limit 50 km/h	96.0%	94.1%	5	1.2×10^{-7}
	End of speed and passing limits	96.8%	76.4%	7	3.6×10^{-9}
	Right-of-way	88.3%	20.2%	16	1.8×10^{-11}
	Speed limit 60 km/h	84.0%	25.9%	18	1.3×10^{-11}
	End of no passing	88.5%	95.1%	20	8.2×10^{-12}
	Children crossing	97.4%	92.3%	24	8.9×10^{-13}
	No vehicle >3.5 tons	62.1%	64.3%	27	3.4×10^{-13}
Speed limit 80km/h	Speed limit 50 km/h	96.9%	99.6%	2	2.2×10^{-3}
	Speed limit 30 km/h	99.9%	100%	3	2.1×10^{-3}
	Speed limit 60 km/h	91.2%	70.9%	4	2.3×10^{-5}
	Yield	86.2%	6.2%	7	5.6×10^{-7}
	Speed limit 20 km/h	98.6%	90.7%	11	3.4×10^{-7}
	No vehicles	74.6%	0%	12	3.3×10^{-7}
	End of Speed limit 80 km/h	90.6%	90.3%	16	2.5×10^{-7}
	No passing	61.9%	9.1%	18	1.7×10^{-7}
	Keep right	95.5%	98.0%	19	1.3×10^{-7}
	Children crossing	94.9%	100%	21	3.6×10^{-8}
	Stop	99.6%	10.9%	22	2.4×10^{-8}
	Slippery road	72.7%	15.4%	28	3.9×10^{-9}
	Bicycles crossing	92.5%	9.6%	31	2.6×10^{-9}
	Narrows right	89.9%	60.4%	32	9.9×10^{-10}

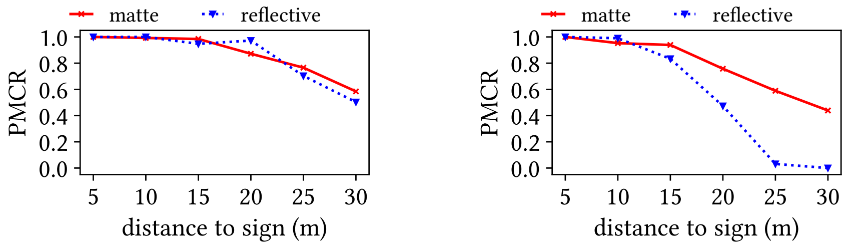
widths and counts. However, it is still notably feasible as it achieves high ASRs on several targeted classes.

5.3.7 Attack Transferability to Unknown Victim Model. To study the dependency on the attacker's knowledge of the victim model, we evaluate attack transferability from a targeted victim model to another model that is unknown and inaccessible to the attacker. Specifically, we test a ResNet-based traffic sign classifier (inaccessible to the attacker) using the traffic sign cropouts collected in the trials in Section 5.3.2, where the attacks were optimized on the AlexNet-based classifier. Figure 14 shows the attack effectiveness on the known (blue) and unknown (red) victim classifiers. Both MR and PMCR decrease when the victim model is unknown to the attacker, as indicated by the overall drops in the red boxes. Interestingly, while the transferability of *GS2* drops significantly in both metrics, *GS1* maintains relatively higher effectiveness. This difference is mainly due to their optimization



(a) MR. (b) PMCR.

Fig. 14. Attack transferability to unknown victim classifier.



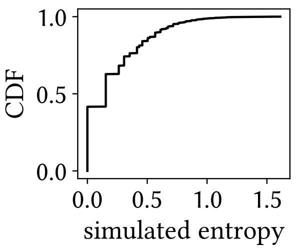
(a) Darker ambient light condition. (b) Brighter ambient light condition.

Fig. 15. Impact of traffic sign texture.

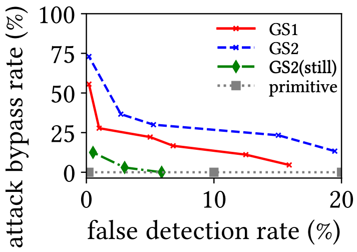
strategies and runtime settings. *GS2* is optimized for a specific target class and operates within a restricted range of offsets, making it harder to achieve the targeted attack when transferred to an unknown model. In contrast, *GS1* allows arbitrary offsets and resulting classes under a non-targeted optimization strategy, providing greater flexibility to achieve untargeted attacks at random offsets and thus exhibiting better transferability to an unknown model. These results suggest that *GS1* is more feasible when the victim model is unknown and inaccessible to the attacker.

5.3.8 Impact of Traffic Sign Texture. Traffic signs may have different plate surface textures (e.g., matte-printed stickers and reflective sheeting) [20, 33]. To investigate the effect of a traffic sign's surface reflectivity on attack performance, we compare two types of "stop" signs: a *matte* sign without reflective coating and one with *reflective* sheeting. We collect traffic sign cropouts while deploying *GS2* at various distances in a stationary setup, alternating between the two sign textures. Figure 15 shows the attack effectiveness against the two textures. Under relatively darker ambient conditions, *GS* achieves similarly high effectiveness on both sign types. However, under brighter conditions, *GS*'s effectiveness decreases more notably on the reflective sign as distance increases. This reduction in effectiveness possibly arises from two factors. First, the reflective sign has a more directional reflection characteristic, causing less attack illumination from the prototyped LED (installed near the sign's bottom edge) to reach the victim camera at greater distances. Second, with bright ambient light, the reflective sign captures and reflects more ambient light, which increasingly overwhelms the relatively dimmer attack illumination at larger distances. These findings suggest that attacking highly reflective traffic signs may require stronger attack setups, such as spotlight projections with higher intensity and reduced angles of incidence.

5.3.9 Effectiveness Against Anomaly Detection. We conduct experiments to understand whether *GS* can keep covert against an entropy-based anomaly detector. We design the detector by profiling the entropy characteristics of the sign classifier on the GTSRB dataset. We generate pseudo-prediction sequences and calculate entropy values in sliding windows of 45 frames. Figure 16(a)



(a) Simulated entropy on the victim classifier on GTSRB.



(b) Comparison of attack bypass rates against anomaly detection.

Fig. 16. Entropy profiling for anomaly detection and attack opportunity against anomaly detection.

shows the CDF of the simulated entropy, which we use to indicate the entropy boundary without attack. In other words, when using an entropy value as the detection threshold, the difference between 1 and the corresponding CDF is likely representative of the false detection rate, which characterizes the unnecessary troubleshooting overhead incurred to the human driver by the detection.

We assume the attack can bypass the anomaly detection only when all sliding windows have entropy values smaller than the threshold. Otherwise, the anomaly may be detected by the system to trigger the fail-safe mechanism. Figure 16(b) shows the attack bypass rates under different false detection rates on attack-free sequences for different methods. When the false detection rate is less than 0.2%, *GS2* achieves 73.3% bypass rate. When the false detection rate is 5.3% and 19.5%, the bypass rate are around 30.3% and 13.4%. *GS1* achieves bypass rates of 55.6%, 27.8%, and 4% when the false detection rate is at 0.2%, 1%, and 15%. On the contrary, other baselines can hardly bypass the anomaly detection. Results show that *GS* has considerable opportunities to bypass the anomaly detection due to its design to achieve stable attack.

5.4 Evaluation on Lab Testbed

We investigate the impacts of various factors on our lab testbed. In this subsection, unless otherwise specified, we plan the attack based on a camera exposure time of $1/1,000$ s and sign-camera distance of 2 m on the testbed, which is equivalent to 20 m in real world.

5.4.1 Exposure Requirement. We use *GS2* to test with exposure time t_{exp} ranging from $1/2,000$ s to $1/250$ s at different sign-camera distances. As shown in Figure 17(a), when t_{exp} is small (i.e., $\leq 1/750$ s), the PMCR is always high across a range of sign-camera distance. When $t_{exp} = 1/500$ s in Figure 17(b), the PMCR is high when the equivalent sign-camera distance is shorter than 17.5 m. When $t_{exp} = 1/250$ s in Figure 17(c), the targeted attack fails at any distance as PMCR is always zero, and MR only remains high within short distances. This is because when t_{exp} is larger, adjacent scanlines have a larger ratio of time overlaps being exposed. With larger t_{exp} or smaller N_{sign}, the colored stripes in a perturbation become more vague and thus, less effective. These results suggest that GS requires short t_{exp} (e.g., $<1/500$ s on AR023ZWDR) at the vehicle camera to ensure successful attacks along a long distance. As autonomous vehicles are highly motion-involved, to freeze the rapid changes in the surrounding environment, a short t_{exp} less than $1/500$ s is usually required to avoid motion blur [1]. Thus, the exposure requirement does not impede GS.

5.4.2 Impact of Exposure Estimation Bias. We use *GS2* to study the tolerance to exposure estimation bias. We prepare the attacks for different exposure times t_{exp}, i.e., $1/750$ s, $1/1,000$ s, $1/1,500$ s,

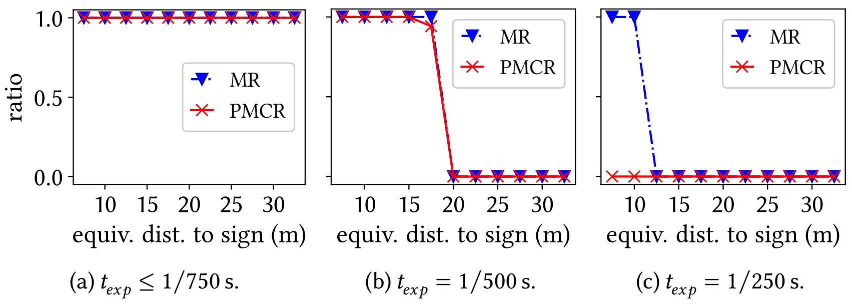


Fig. 17. Effectiveness vs. sign-camera distance under different t_{exp}.

and $1/2,000$ s. Then, we test them with different actual t_{exp} on the victim camera. Figure 18 shows the PMCR under exposure estimation bias. All four attack exposure settings perform well within wide ranges of the actual exposure, showing the robust attack effectiveness against exposure bias. The exposure bias can affect the differences between the desired and actual perturbation sharpness, size and the overall image brightness. First, when the actual t_{exp} is larger than $1/500$ s, the attack PMCR is low due to the poor perturbation sharpness. Second, the perturbation size defined by the duration attack window is affected by the bias in t_{exp}. When the actual t_{exp} is within the working range (i.e., $< 1/500$ s), as the t_{exp} is already small, the introduced size error is usually small and tolerable. Third, camera exposure affects the amount of input light, resulting in differences in image brightness between training data and run-time images. Large mismatches in exposure may cause large brightness difference and reduce the attack effectiveness.

5.4.3 Impact of Lighting Conditions. As it is hard to control the ambient light outdoors, we use controllable light sources indoors to study the relationship between the attack performance and lighting conditions. We use two studio lamps to change the ambient light level to mimic different light levels outdoors. Figure 19 shows the attack effectiveness under different ambient lighting conditions measured on the traffic signs with reference to outdoor conditions [47]. With stronger ambient light, the attack performance decreases. This degradation occurs because the attack light is overwhelmed by the ambient light. Therefore, with brighter ambient light, the attack light needs higher power. Besides, this suggests that the attacker may need to consider the time and location when planning the attack, e.g., avoid those where direct sunlight shines on the sign (usually over 100,000 lux). Note that in Section 5.3, we have demonstrated the attack effectiveness of *GS* under normal daytime ambient light conditions.

5.4.4 More Results on Another Victim Classifier and Sign Colors. We evaluate the feasibility and generality of *GS* by testing it with another victim classifier and additional traffic sign classes featuring different colors. Specifically, we use a ResNet-based classifier as the victim model and include another two representative traffic signs, "priority road" and "ahead only", which have yellow and blue as their primary colors, respectively. Table 3 summarizes the attack effectiveness against the ResNet-based classifier and various traffic sign colors. The results demonstrate that *GS* is feasible against different victim models and traffic sign colors.

5.4.5 Effectiveness Against Another Camera and Impact of Focal Length. We use the Arducam AR1820HS mobile camera and *GS2* to demonstrate attack effectiveness across different cameras and to study the impact of camera focal length Z_f. During the test, we switched the lens of the AR1820HS, using focal lengths of 3.9 mm, 6 mm, 12.5 mm, and 25 mm. Figure 20 shows the PMCR under different focal length settings. Beyond the Leopard Imaging AR023ZWDR evaluated earlier,

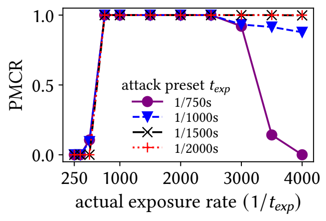


Fig. 18. Effectiveness with exposure bias.

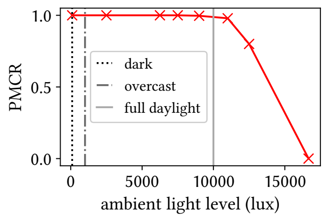


Fig. 19. Effectiveness vs. ambient light.

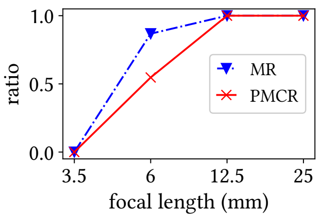


Fig. 20. Effectiveness vs. focal length (Arducam AR1820HS).

Table 3. *GS*'s Effectiveness Against ResNet-based Classifier and Different Sign Colors

Original	**MR for GS1**		**Target of GS2**	**PMCR for GS2**	
	WB	**BB**		**WB**	**BB**
Stop	100%	99.2%	Priority road	100%	0%
			Speed limit 30 km/h	93.2%	36.9%
			Double curve	90.7%	86.5%
			Speed limit 20 km/h	63.2%	0%
Yield	27.3%	0%	Speed limit 30 km/h	97.8%	0%
			Priority road	91.4%	0%
Speed limit 30km/h	71.6%	36.7%	Stop	100%	85.8%
			Speed limit 70 km/h	100%	100%
			Speed limit 80 km/h	95.8%	100%
			Priority road	93.4%	96.6%
			No vehicles	81.0%	85.8%
			Speed limit 50 km/h	75.7%	30.5%
			Go straight or right	74.1%	43.2%
Speed limit 80km/h	100%	80.8%	Speed limit 60 km/h	100%	100%
			Speed limit 50 km/h	89.6%	98.4%
			Speed limit 30 km/h	78.1%	0%
Priority road	82.7%	70.5%	Stop	100%	90.5%
			No entry	100%	100%
			Bicycles crossing	99.3%	5.9%
Ahead only	91.5%	82.4%	Go straight or right	100%	83.2%
			Turn right ahead	99.6%	94.1%
			Children crossing	99.4%	0%

GS remains effective against the Arducam AR1820HS. However, the attack performance decreases as the focal length becomes smaller. This is because a smaller Z_f results in a smaller N_{sign}, making the stripe patterns less distinguishable, as discussed in the impact of distance in Section 5.3.4. These results suggest that cameras with larger focal lengths are more vulnerable to *GS*.

6 Countermeasures

In this section, we first propose ***GhostBuster***, a software-based front-end add-on defense module designed to detect threats from GhostStripe and restore clean traffic sign images from compromised ones. Additionally, we discuss other possible countermeasures on the level of hardware and autonomous driving system.

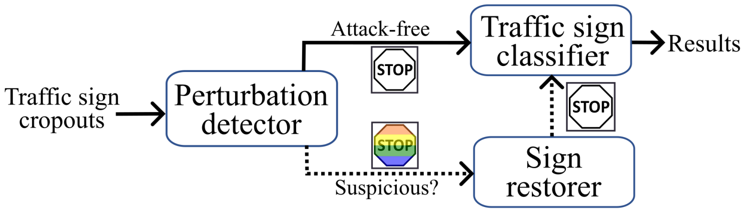


Fig. 21. GhostBuster's defense pipeline.

6.1 GhostBuster Design

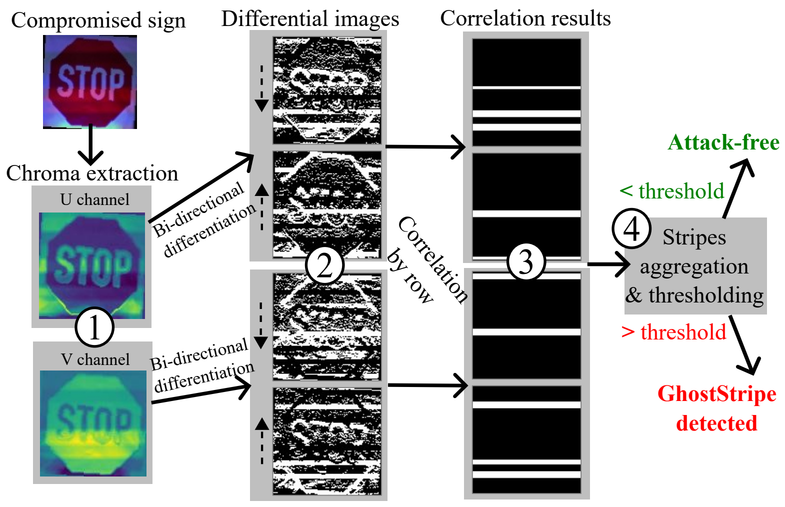
Figure 21 provides an overview of the GhostBuster's defense strategy. GhostBuster acts as a front-end add-on module for the traffic sign classifier and includes a perturbation detector and a sign restorer to counteract GhostStripe. Under normal conditions, when no threat from GhostStripe is detected by the perturbation detector, the traffic sign cropout is fed directly to the traffic sign classifier. If the detector identifies a potential GhostStripe compromise, the traffic sign cropout is processed by the sign restorer to eliminate the colorful and stripy perturbation before being sent to the classifier for inference.

6.1.1 Perturbation Detector. In this section, we explore two methods (i.e., image characteristic analysis and deep learning) to detect GhostStripe's perturbations in the traffic sign cropouts.

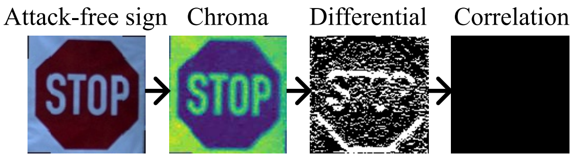
Pattern correlation-based detection. We propose a detection scheme by analyzing the special image characteristic caused by GhostStripe. The workflow is shown in Figure 22(a). In the GhostStripe-affected images, we observe that at the borders between adjacent color stripes, pixel values in rows in the chroma channels (U and V in the YUV color model, as shown in Figure 22(a)-①) exhibit continuous ascending or descending trends. Then, we compute vertical differential images in both up-down and down-up directions, and binarize these images by distinguishing between positive and non-positive values, as shown in Figure 22(a)-②. Note that, we employ differential operation in two vertical directions and keeping the positive values, rather than solely relying on one direction and taking the absolute value. This is to mitigate noise (as shown as the scattered white points in Figure 22(a)-②) aggregation across the entire image when using the latter method. After that, we perform a 1-D correlation along the vertical axis on the binarized differential images with a horizontal all-1-row to detect the stripe boundaries, indicated as the white stripes in the correlation results in Figure 22(a)-③. In contrast, the chroma of an attack-free traffic sign is relatively uniform in the vertical direction, and thus hardly generates stripy patterns after the processing, as shown in Figure 22(b). By aggregating and thresholding the correlation results, the detector suggests that the input sign cropout is compromised by GhostStripe or not.

Deep learning-based detection. Another possible way to detect the attack is to train a deep learning-based classifier to distinguish the traffic signs compromised by GhostStripe. We employ a convolution neural network, which comprises two convolutional layers, each followed by ReLU activation and max-pooling operations. Subsequently, the feature maps are flattened and fed into two fully connected layers, culminating in a final layer that produces binary class predictions, inferring the input image is compromised by GhostStripe or not.

6.1.2 Sign Restorer. The objective of the sign restorer is to remove GhostStripe-induced color stripes from the traffic sign cropout, thereby reconstructing the natural appearance of the captured traffic sign for classification. Although invisible in the real world, GhostStripe's perturbations can drastically distort the compromised traffic signs in captured images, making restoration challenging. Therefore, we consider the restoration as an image-to-image translation problem, transforming



(a) Pattern correlation-based detection against GhostStripe.



(b) Detection on an attack-free traffic sign.

Fig. 22. Pattern correlation-based perturbation detection.

images with colorful and horizontally stripy distortions into relatively uniform images with limited and natural colors and distinct sign contours.

We employ *CycleGAN* [61] as the restorer. CycleGAN uses a **Generative Adversarial Network** (**GAN**) architecture designed for unpaired image-to-image translation. It employs two generator-discriminator pairs: one for translating images from the source domain (GhostStripe-affected signs) to the target domain (clean signs), and another for the reverse translation. The network also uses cycle-consistency loss to ensure that an image translated to the target domain and then back to the source domain remains unchanged, preserving the structural integrity of the original image. This dual GAN approach allows CycleGAN to learn the mapping between the two domains, maintaining the underlying structure of the traffic sign while modifying the surface appearance to remove the perturbations. Through this adversarial process, the generator learns to produce more realistic and accurate restorations. The trained generator for translating from the source domain to the target domain then serves as the restorer.

6.2 GhostBuster Implementation and Evaluation

6.2.1 Datasets. We use the following two datasets to train (or profile) and test the detector and restorer in GhostBuster: (1) **Synthesized dataset:** We simulate GhostStripe-affected traffic signs using attack-free traffic signs collected from our outdoor testbed. This is achieved by overlaying various color stripes under different camera and attack settings, including varying camera exposure rates, sign sizes in the images, and stripe counts. The dataset comprises 400 attack-free images and 400 GhostStripe-affected images, divided into training, validation, and testing subsets with ratios of

Table 4. Performance of the Proposed Attack Detection Methods

Detection methods	**Synthesized**		**Real-world**	
	TPR	**FPR**	**TPR**	**FPR**
Pattern correlation-based	96.6%	0%	95.9%	1.5%
Deep learning-based	100%	0%	73.8%	0%

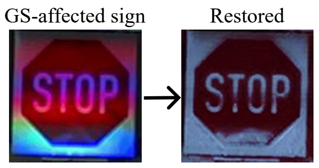
60%, 20%, and 20%, respectively. The former two subsets are used for profiling the parameters in pattern correlation-based detector, and training the deep-learning based detector and restorer. (2) **Real-world dataset:** we also utilize traffic sign images collected from our real-road evaluation in Section 5 as a real-world testing set.

6.2.2 Implementation and Evaluation. **Implementation of perturbation detectors.** (1) For the pattern correlation-based method, we empirically set the correlation output threshold at 0.85 and consider images compromised by GhostStripe if the total rows of detected stripes in the four differential images exceed 10% of the image height. To mitigate the influence of "intrinsic stripes" from long horizontal borders of some traffic signs, we discard the top and bottom 20% of the correlation results. (2) For the deep learning-based method, we train the network with cross-entropy loss and Adam optimizer at a learning rate of 10^{-3} for 10 epoches.

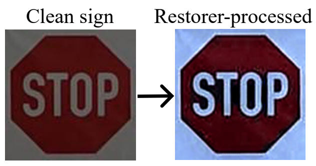
Implementation of sign restorer. We follow the implementation of *CycleGAN* [61] with a generator architecture based on ResNet with 9 residual blocks, and a discriminator architecture based on a 70×70 PatchGAN [21]. We train the *CycleGAN* with Adam optimizer with an initial learning rate of 2×10^{-4} for 100 epoches, and linearly decaying learning rates to zero for another 100 epochs.

Performance of perturbation detectors. Table 4 summarizes the performance of the detection methods. On the synthesized dataset, the pattern correlation-based detection achieves a 96.6% **true positive rate** (**TPR**) on GhostStripe-affected images and a 0% **false positive rate** (**FPR**) on attack-free images. On the real-world outdoor testbed dataset, the TPR and FPR are 95.9% and 1.5%, respectively. The deep learning-based detection achieves a 100% TPR and 0% FPR on the synthesized dataset. However, when tested with real-world images, the TPR drops to 73.8%. The observed performance disparity in deep learning-based method can be attributed to the inherent differences between the synthesized training data and real-world scenarios. This shows the fundamental characteristic of data-driven approaches: their efficacy is profoundly influenced by the representativeness of the training data. However, collecting sufficient and diverse real-world data under various attack conditions can be labor-intensive. The pattern correlation-based method, on the other hand, captures the underlying characteristics of the attack and demonstrates better generalizability in real-world scenarios compared to the deep learning-based method. Therefore, we adopt the pattern correlation-based one as our perturbation detector.

Performance of sign restorer. Figure 23 illustrates the effects of the sign restorer. As shown in Figure 23(a), the GhostStripe-induced distortion is significantly suppressed. When testing on real-world images affected by GhostStripe perturbations that lead to misclassification, 70.8% of the restored images were correctly classified as their original classes by the traffic sign classifier. While the classifier's performance is not fully recovered (likely due to minor residual distortions), this demonstrates the restorer's effectiveness in mitigating the impact of GhostStripe and enhancing the recognizability of compromised traffic signs. Furthermore, to account for false positives from the perturbation detector, we evaluated the restorer's impact on real-world attack-free images. As shown in Figure 23(b), the restorer only alters the color tint of the traffic sign, while the sign still appear natural. When tested with the classifier, the restorer-processed clean images were 100% correctly classified. This indicates that this minimal alteration did not affect the classifier's ability

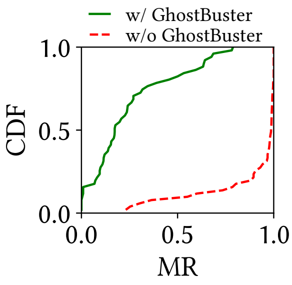


(a) GS-affected sign.

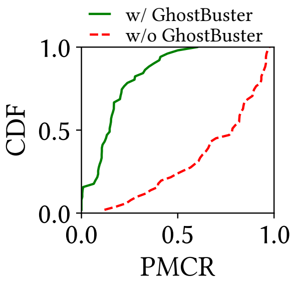


(b) Attack-free sign.

Fig. 23. The processing effect of the restorer on GhostStripe-affected and attack-free traffic signs.



(a) MR.



(b) PMCR.

Fig. 24. Comparisons of GhostStripe's effectiveness on real-world trials with and without GhostBuster.

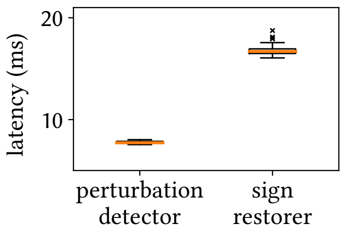


Fig. 25. Latencies of GhostBuster on Jetson Orin.

to correctly identify attack-free traffic signs, and the restorer preserves the integrity of already clean images.

Overall performance of GhostBuster. We test the overall performance of GhostBuster on the frame sequences collected in the driving trials in Section 5. We execute GhostBuster across all the frames, i.e., run the pattern correlation-based detector, restore the clean sign from the perturbation-detected frames, and check the classification results on the GhostBuster-rectified frame sequences. Figure 24 compares the MR and PMCR with and without GhostBuster. Results show that GhostBuster can largely reduce the attack effectiveness of GhostStripe. For example, in 5.8% of the trials, the MR is 0 (100% correct classification and GhostStripe are totally defensed); in 27.4% and 52.9% of the trials, the MR is less than 10% and 20%, respectively. Future work improving the restorer's performance (e.g., improving the image-to-image translation model/training process, generating more representative training datasets) will be meaningful.

Computational overhead of GhostBuster. We implement GhostBuster on an NVIDIA Jetson AGX Orin at a 50 W power setting, which is commonly used as a central computing platform on autonomous vehicles. Figure 25 presents the latency measurements of GhostBuster's two modules. The pattern correlation-based perturbation detector has a mean latency of 7.7 ms. Given a typical camera frame rate of 30 FPS (i.e., frame interval of 33 ms), the detector supports real-time processing at 30 FPS. It can be executed regularly (i.e., every few frames or even per frame), when a traffic sign is detected and needs to be recognized. The sign restorer requires more time, with a mean latency of 16.7 ms. When the presence of GhostStripe is detected, the total time overhead for running both modules is 24.4 ms. Upon detecting GhostStripe, the autonomous driving system can alert the human driver. Before the driver takes over, the sign restorer temporarily recovers the clean traffic sign for recognition, allowing the vehicle to continue autonomous operation. Because a traffic sign's meaning remains constant, the restorer can be executed every few frames to conserve computational resources. This strategy allows GhostBuster to counter the threat without exhausting computational resources, while still engaging a human driver when necessary. Additional measures, such as selecting appropriate computing hardware and further optimizing the defense implementation, can also be employed to ensure GhostBuster meets real-time requirements.

6.3 Other Hardware/System-level Enhancement

There are several other countermeasures that may be applied to counteract the GhostStripe attack.

Camera exposure mechanism. A straightforward way is to replace the widely used rolling shutter cameras by global shutter cameras. Another countermeasure is to shuffle or randomize the sequence of scanline exposure [16, 48], which spreads the attack pattern to various scanlines different from the desired perturbation. However, such countermeasures impose new requirements and extra costs on the manufacturers of autonomous vehicles and cameras, and may not be feasible for all autonomous vehicles.

System-level redundancy. Multi-camera coordination may help mitigate the attack effect. Since GhostStripe targets a single camera, it is usually ineffective against other cameras with different specifications. However, many autonomous vehicle systems use hierarchical camera coordination. For instance, Baidu Apollo prioritizes the telephoto camera for traffic light recognition and uses a wide-angle camera as a backup [5]. In this case, attackers can still focus on the main camera. This strategy is further supported by results in Section 5.4.5, which reveal that telephoto cameras (with larger focal lengths) are more vulnerable to GhostStripe. Another countermeasure is using **High-Definition** (**HD**) maps to assist traffic sign perception, providing semantics and locations of traffic signs. However, creating, updating, and scaling HD maps and labeling all traffic signs is costly and time-consuming [4], reducing the desirability of this approach. Moreover, maps may not cover all areas, and may not adapt to changes in traffic signs due to *ad hoc* construction or special events.

7 Attack against Traffic Sign Detector

While we focus on compromising the traffic sign classifier in this article, we also explore GhostStripe's attack effectiveness against the front-end traffic sign detector, abbreviated as ***GS-D***. We use the YOLOv5s object detector [24] as the victim detector model and "stop" as the sign to be compromised.

7.1 Attack Vector Optimization

The overall design against the detector is similar to the one against the classifier, despite the differences in the attack vector optimization. The optimization workflow is summarized in Figure 26. In *GS-D*, one more step is to overlay the compromised traffic sign in the FoV as the victim model input, and uses a different loss function. Specifically, we collect the attack-free camera FoV $I_{fov,amb}$ when the victim camera approaches the attack-free traffic sign. Then, we use the victim detector to detect the groundtruth location (u_{sign}, v_{sign}), width, and height (W_{sign}, N_{sign}) of the traffic sign $I_{sign,amb}$ within each attack-free FoV $I_{fov,amb}$. Each $I_{sign,amb}$ cropout undergoes a brightness enhancement

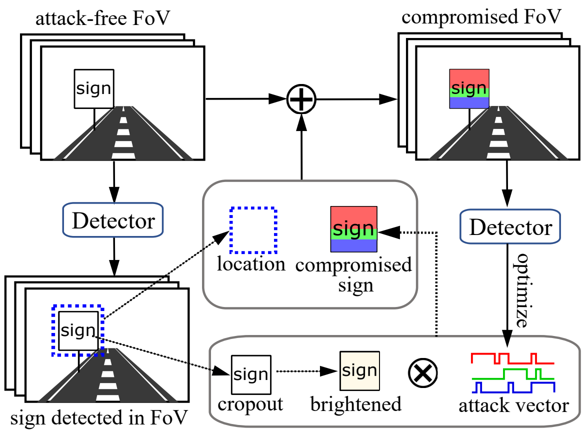


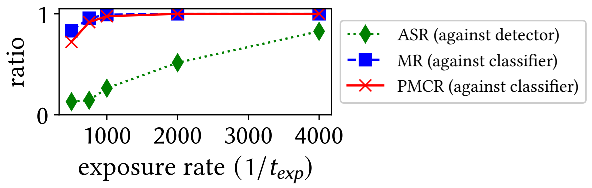
Fig. 26. Workflow of attack optimization against traffic sign detector.

process to mimic the effect of full-intensity LED lighting, transforming it into $I_{sign,full}$, to obtain the synthesized $I_{sign,att}$ by substracting $I_{sign,full}$ by $I_{sign,amb}$. We follow the training procedure described in Section 4.3 to optimize the $f_0(t)$. In each training frame, according to the actual N_{sign}, the $f_0(t)$ will be upscaled to $f(t)$ and simulate the compromised traffic sign I^{ϕ}_{sign}. Subsequently, within each FoV image $I_{fov,amb}$, the attack-free traffic sign is replaced by the compromised version. This is achieved by overlaying I^{ϕ}_{sign} onto the corresponding region within $I_{fov,amb}$ at the ground truth location (u_{sign}, v_{sign}). The resulting image, denoted as I^{ϕ}_{fov}, represents the FoV under attack, and is fed into the detector for attack optimization. The optimization goal is to jointly decrease the class and IoU prediction performance of the detector, by solving $\text{argmax}_{f_0(t)}\ \mathbb{E}_{\phi}[\ell_{cls}(\mathcal{D}(I^{\phi}_{fov}), j) \times \ell_{obj}(\mathcal{D}(I^{\phi}_{fov}), U)]$, where $\mathcal{D}(\cdot)$ is the detector, j is the groundtruth class, U is the groundtruth IoU, $\ell_{cls}(\mathcal{D}(I(u,v)), j)$ is the classification loss for the groundtruth class j, $\ell_{obj}(\mathcal{D}(I(u,v)), U)$ is the objectness loss (IoU prediction loss) for the groundtruth IoU, when the detector is fed with $I(u,v)$.

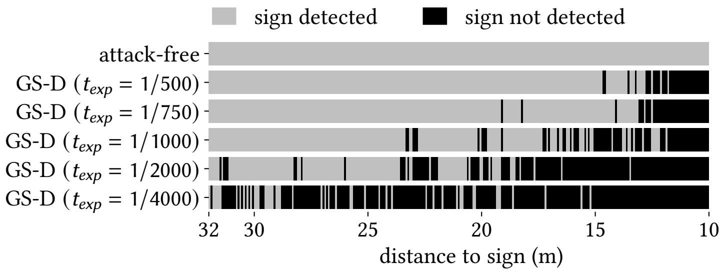
7.2 Proof-of-Concept Evaluation

We evaluate GhostStripe against the detector in simulation using *GS2* under a white-box setting. We prepare attack vectors for different victim camera exposure times. The testing data is synthesized from video clips (different from the training data) taken by the victim camera approaching the traffic sign, following the process described in Section 7.1 and Figure 26. We use the default confidence threshold in YOLOv5s (0.25) for successful detection. We define the ASR of *GS-D* as the ratio of frames where the traffic sign is not successfully detected under attack to the total frames.

Figure 27(a) presents the ASR of *GS-D* against the detector at different exposure rate. The overall attack effectiveness of *GS-D* is lower than GhostStripe against the classifier. In Figure 27(b), we also visualize the attack results across frames when the camera approaches the traffic sign. The impact of distance shows a similar trend with the classifier as discussed in Section 5.3.4, as the attack is more effective at shorter distances. Besides, the shorter the exposure time, the more effective the attack. This is because when t_{exp} is larger, adjacent scanlines have a larger ratio of time overlaps being exposed. With larger t_{exp} or smaller N_{sign} (due to longer distance), the colored stripes become more vague and thus less effective. Compared to *GS* against the classifier, *GS-D* requires shorter t_{exp} and distance to be effective. This further indicates that spoofing the detector on the global FoV input requires more prominent perturbations (sharper stripes) on the traffic sign.



(a) ASR of *GS-D* under different exposure settings.



(b) Example of attack results against the detector on the consecutive frames when the camera is approaching the traffic sign.

Fig. 27. Results on *GS-D*.

8 Limitations and Discussions

Physical access for sniffer installation. The requirement of physical access for sniffer installation may limit GhostStripe2's opportunity. A determined adversary could potentially obtain the physical access by collaborating with an auto-care provider for installation. Alternatively, attackers may resort to GhostStripe1 for untargeted attacks. Exploring real-time remote sensing or eavesdropping for camera operation is an interesting future work direction.

Attacker's prior knowledge of the victim vehicle. Although Section 5.3.7 shows that *GS1* can transfer attack effectiveness to an inaccessible victim DNN, it remains preferable for the attacker to have black-box access or white-box knowledge to carry out more controllable and efficient attacks. GhostStripe requires knowledge of the victim camera's parameters, which can typically be obtained through datasheets or reverse engineering. However, if such specifications are unavailable, it becomes challenging for GhostStripe to launch effective attacks, as the system customizes the light signal modulation based on the camera's operation.

Attack practicability under different conditions. Our prototype achieves similar scales as prior works [10, 22, 54] and show high attack chances. For longer ranges and stronger ambient light conditions, the attacker may need to adopt brighter LEDs and more efficient light projection setups. For very high victim vehicle speed, the system latencies (e.g., from vehicle tracker and camera sniffer to the LED controller) may need to be further reduced.

Autonomous driving system-level evaluation. It is interesting to understand whether the misled traffic sign recognition results, which may not be fully stable, can lead to safety incidents. Simulations are likely the safest method to study this. However, to the best of our knowledge, publicly available driving agents only handle traffic lights, not traffic signs sensed at run time. Future work addressing this gap, which requires the construction of a full-fledged publicly accessible driving agent, is meaningful.

Other car-borne cameras. In the real-road implementation, we evaluate the Leopard Imaging AR023ZWDR camera because it is the default main camera used in Baidu Apollo autonomous driving system [7] and the only one used for vehicles. We also demonstrate the attack on the Arducam

AR1820HS camera in Section 5.4.5, further supporting the generality of the attack across rolling shutter cameras. Evaluating the proposed attack against more car-borne cameras is of great interest.

GhostBuster's false alarms due to unintentional light interference. Flickering ambient light can also induce RSE and produce color stripes in the camera's FoV, potentially triggering false alarms in GhostBuster's perturbation detector. However, it has been revealed that unintentional RSE stripes may also degrade image-based recognition [29]. Therefore, GhostBuster may also help enhance the reliability of traffic sign recognition under such cases. Note that in our urban site survey, we have not observed RSE stripes on traffic signs induced by flickering road lights. In rare scenarios where ambient flickering lights do interfere with imaging, front-end de-striping methods or stripe-robust perception models may be needed to maintain normal perception functionality, and prevent frequent execution of GhostBuster's restorer.

GhostBuster's effectiveness against other types of attacks. GhostBuster is designed specifically to defend against the GhostStripe attack. Its design assumes colored stripy perturbations applied to traffic signs. As a result, GhostBuster is mainly effective against GhostStripe (and possibly other attacks that create similar stripe-based perturbations, e.g., through projection or translucent films). While its detect-and-recover principle may inspire broader defenses, GhostBuster is not a universal solution to all adversarial perturbations. Future work may extend this framework to more physical adversarial attacks with different perturbation characteristics.

9 Conclusion

This article presents GhostStripe, an attack system that exploits the RSE of CMOS cameras to generate adversarial stripes that mislead traffic sign recognition in autonomous vehicles. To achieve a stable attack, GhostStripe controls the timing of the LED's modulated light emission to adapt to the camera's operations and the victim vehicle's movement. In our experiments, we demonstrate that GhostStripe can consistently spoof traffic sign recognition, producing semantically conflicting results across consecutive frames. Such attacks could trigger life-threatening consequences for the victim vehicle and its surrounding environment. To counteract this attack, we propose GhostBuster, a defense module designed to detect and mitigate the threats posed by GhostStripe. Additionally, we discuss other potential countermeasures at the levels of camera sensor and autonomous driving system.

Acknowledgments

We thank Yimin Dai for assisting with the exploration of vehicle tracking solutions, Junming Zeng for helping customize the LED drivers, and Changhao Tian for driving the car in the real-world experiments.

References

[1] 2012. *When to Use Different Shutter Speeds (A Complete List)*. Retrieved 22 Nov. 2023 from https://expertphotography.com/when-to-use-different-shutter-speeds/

[2] 2018. *What are the Most Common Traffic Signs?* Retrieved 20 Oct. 2023 from https://topdriver.com/education-blog/what-are-the-most-common-traffic-signs/

[3] 2020. *Teardown: Teslaś Hardware Retrofits for Model 3*. Retrieved 24 June 2023 from https://www.eetasia.com/teslas-hardware-retrofits-for-model-3/

[4] 2021. *The Road to Everywhere: Are HD Maps for Autonomous Driving Sustainable?* Retrieved 16 Nov. 2023 from https://www.autonomousvehicleinternational.com/features/the-road-to-everywhere-are-hd-maps-for-autonomous-driving-sustainable.html

[5] 2022. *Apollo Traffic Light Perception*. Retrieved 1 June 2023 from https://github.com/ApolloAuto/apollo/blob/master/docs/06_Perception/traffic_light.md

[6] 2022. *Manual on Uniform Traffic Control Devices for Streets and Highways*. Retrieved 1 Jan. 2023 from https://mutcd.fhwa.dot.gov/

[7] 2023. *Apollo hardware development platform*. Retrieved 1 June 2023 from https://developer.apollo.auto/platform/hardware.html

[8] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *ICML*. PMLR, 284–293.

[9] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. arXiv:1712.09665. Retrieved from https://arxiv.org/abs/1712.09665

[10] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *IEEE SP*. IEEE, 176–194.

[11] Nicholas Carlini. 2023. *A Complete List of All (arXiv) Adversarial Example Papers*. Personal website. Retrieved from https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[12] Christos Danakis, Mostafa Afgani, Gordon Povey, Ian Underwood, and Harald Haas. 2012. Using a CMOS camera sensor for visible light communication. In *IEEE Globecom Workshops*. IEEE, 1244–1248.

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *IEEE/CVF CVPR*. 1625–1634.

[14] GETCAMERAS. 2020. *Rolling Versus Global Shutter*. Retrieved 24 Sep. 2022 from https://www.get-cameras.com/FAQ-ROLLING-VS-GLOBAL-SHUTTER

[15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv:1412.6572. Retrieved from https://arxiv.org/abs/1412.6572

[16] Jinwei Gu, Yasunobu Hitomi, Tomoo Mitsunaga, and Shree Nayar. 2010. Coded rolling shutter photography: Flexible space-time sampling. In *ICCP*. IEEE, 1–8.

[17] Dongfang Guo, Yuting Wu, Yimin Dai, Pengfei Zhou, Xin Lou, and Rui Tan. 2024. Invisible optical adversarial stripes on traffic sign against autonomous vehicles. In *ACM MobiSys*. 534–546.

[18] Pengfei Hu, Parth H. Pathak, Xiaotao Feng, Hao Fu, and Prasant Mohapatra. 2015. Colorbars: Increasing data rate of led-to-camera communication using color shift keying. In *CoNEXT*. 1–13.

[19] Wenjun Hu, Hao Gu, and Qifan Pu. 2013. Lightsync: Unsynchronized visual communication over screen-camera links. In *MobiCom*. 15–26.

[20] Identimark. 2023. *Understanding Reflective vs. Non-Reflective Signage*. Retrieved 12 Feb. 2023 from https://identimark.com/news/when-do-i-use-reflective-vs-non-reflective-signage?

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *IEEE CVPR*. 1125–1134.

[22] Xiaoyu Ji, Yushi Cheng, Yuepeng Zhang, Kai Wang, Chen Yan, Wenyuan Xu, and Kevin Fu. 2021. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. In *IEEE SP*. IEEE, 160–175.

[23] Pengfei Jing, Qiyi Tang, Yuefeng Du, Lei Xue, Xiapu Luo, Ting Wang, Sen Nie, and Shi Wu. 2021. Too good to be safe: Tricking lane detection in autonomous driving with crafted perturbations. In *USENIX Security*. 3237–3254.

[24] Glenn Jocher. 2020–. *YOLOv5 by Ultralytics*. Retrieved 1 Jan. 2023 from https://github.com/ultralytics/yolov5

[25] Sebastian Köhler, Giulio Lovisotto, Simon Birnbach, Richard Baker, and Ivan Martinovic. 2021. They see me rollin‘: Inherent vulnerability of the rolling shutter in cmos image sensors. In *ACSAC*. 399–413.

[26] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. 2014. Luxapose: Indoor positioning with mobile phones and visible light. In *MobiCom*. 447–458.

[27] Hui-Yu Lee, Hao-Min Lin, Yu-Lin Wei, Hsin-I Wu, Hsin-Mu Tsai, and Kate Ching-Ju Lin. 2015. Rollinglight: Enabling line-of-sight light-to-camera communications. In *MobiSys*. 167–180.

[28] Haoliang Li, Yufei Wang, Xiaofei Xie, Yang Liu, Shiqi Wang, Renjie Wan, Lap-Pui Chau, and Alex C. Kot. 2020. Light can hack your face! black-box backdoor attack on face recognition systems. arXiv:2009.06996. Retrieved from https://arxiv.org/abs/2009.06996

[29] Ziwei Liu, Tianyue Zheng, Chao Hu, Yanbing Yang, Yimao Sun, Yi Zhang, Zhe Chen, Liangyin Chen, and Jun Luo. 2022. CORE-lens: Simultaneous communication and object recognition with disentangled-GAN cameras. In *ACM MobiCom*. 172–185.

[30] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. 2021. Slap: Improving physical adversarial examples with short-lived adversarial perturbations. In *USENIX Security 21*. 1865–1882.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

[32] Yanmao Man, Ming Li, and Ryan Gerdes. 2020. GhostImage: Remote perception attacks against camera-based image classification systems. In *RAID*. 317–332.

[33] maneki signage. 2025. *Traffic signs for car parks and roadworks*. Retrieved from https://www.maneki-signage.sg/online-catalog/traffic-signs

[34] Natalia D. Mankowska, Anna B. Marcinkowska, Monika Waskow, Rita I. Sharma, Jacek Kot, and Pawel J. Winklewski. 2021. Critical flicker fusion frequency: A narrative review. *Medicina* 57, 10 (2021), 1096.

[35] Enrique Marti, Miguel Angel De Miguel, Fernando Garcia, and Joshue Perez. 2019. A review of sensor technologies for perception in automated driving. *IEEE ITS Magazine* 11, 4 (2019), 94–108.

[36] Jonas Mockus. 2005. The Bayesian approach to global optimization. In *IFIP*. Springer.

[37] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. 2020. Phantom of the ADAS: Securing advanced driver-assistance systems from split-second phantom attacks. In *ACM CCS*. 293–308.

[38] Fernando Nogueira. 2014. *Bayesian Optimization: Open Source Constrained Global Optimization Tool for Python*. GitHub. Retrieved from https://github.com/fmfn/BayesianOptimization

[39] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. 2015. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. In *BlackHat Europe 11*.

[40] Niranjini Rajagopal, Patrick Lazik, and Anthony Rowe. 2014. Visual light landmarks for mobile devices. In *IPSN*. IEEE, 249–260.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *IEEE CVPR*. 779–788.

[42] Takami Sato, S. Hrushikesh Bhupathiraju, Michael Clifford, Takeshi Sugawara, Qi Alfred Chen, and Sara Rampazzi. 2024. Invisible reflections: Leveraging infrared laser reflections to target traffic sign perception. In *NDSS*.

[43] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. 2021. Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In *USENIX Security*. 3309–3326.

[44] Athena Sayles, Ashish Hooda, Mohit Gupta, Rahul Chatterjee, and Earlence Fernandes. 2021. Invisible perturbations: Physical adversarial examples exploiting the rolling shutter effect. In *IEEE/CVF CVPR*. 14666–14675.

[45] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM CCS*. 1528–1540.

[46] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2011. The German traffic sign recognition benchmark: A multi-class classification competition. In *IJCNN*. IEEE, 1453–1460.

[47] The Engineering ToolBox. 2004. *Illuminance - recommended light levels*. Retrieved 1 Oct. 2023 from https://www.engineeringtoolbox.com/light-level-rooms-d_708.html

[48] Esteban Vera, Felipe Guzmán, and Nelson Díaz. 2022. Shuffled rolling shutter for snapshot temporal imaging. *Optics Express* 30, 2 (2022), 887–901.

[49] Wei Wang, Yao Yao, Xin Liu, Xiang Li, Pei Hao, and Ting Zhu. 2021. I can see the light: Attacks on autonomous vehicles using invisible lights. In *ACM CCS*. 1930–1944.

[50] Hui Wei, Hao Tang, Xuemei Jia, Zhixiang Wang, Hanxun Yu, Zhubo Li, Shin'ichi Satoh, Luc Van Gool, and Zheng Wang. 2022. Physical adversarial attack meets computer vision: A decade survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 12 (2024), 9797–9817.

[51] Xingxing Wei, Bangzheng Pu, Jiefan Lu, and Baoyuan Wu. 2022. Visually adversarial attacks and defenses in the physical world: A survey. arXiv:2211.01671. Retrieved from https://arxiv.org/abs/2211.01671

[52] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. 2020. Adversarial t-shirt! evading person detectors in a physical world. In *ECCV*. Springer, 665–681.

[53] Chen Yan, Wenyuan Xu, and Jianhao Liu. 2016. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. *Def Con* 24, 8 (2016), 109.

[54] Chen Yan, Zhijian Xu, Zhanyuan Yin, Stefan Mangard, Xiaoyu Ji, Wenyuan Xu, Kaifa Zhao, Yajin Zhou, Ting Wang, Guofei Gu, et al. 2022. Rolling colors: Adversarial laser exploits against traffic light recognition. In *USENIX Security*. 1957–1974.

[55] Yanbing Yang, Jie Hao, and Jun Luo. 2017. CeilingTalk: Lightweight indoor broadcast through LED-camera communication. *IEEE TMC* 16, 12 (2017), 3308–3319.

[56] Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu. 2015. Towards real-time traffic sign detection and classification. *IEEE TITS* 17, 7 (2015), 2022–2031.

[57] Yanbing Yang and Jun Luo. 2019. Composite amplitude-shift keying for effective LED-camera VLC. *IEEE TMC* 19, 3 (2019), 528–539.

[58] Lan Zhang, Cheng Bo, Jiahui Hou, Xiang-Yang Li, Yu Wang, Kebin Liu, and Yunhao Liu. 2015. Kaleido: You can watch it but cannot record it. In *MobiCom*. 372–385.

[59] Xiao Zhang, Griffin Klevering, Juexing Wang, Li Xiao, and Tianxing Li. 2023. Rofin: 3d hand pose reconstructing via 2d rolling fingertips. In *MobiSys*. 330–342.

[60] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2020. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *IEEE/ACM ICSE*. IEEE, 347–358.

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE ICCV*. 2223–2232.

[62] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating visual privacy protection using a smart led. In *MobiCom*. 329–342.

[63] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. 2016. Traffic-sign detection and classification in the wild. In *IEEE CVPR*. 2110–2118.

Received 10 August 2024; revised 13 October 2025; accepted 18 October 2025