PhyScout: Detecting Sensor Spoofing Attacks via Spatio-temporal Consistency

Yuan Xu
Nanyang Technological University
College of Computing and Data Science
Sigapore
xu.yuan@ntu.edu.sg

Gelei Deng
Nanyang Technological University
College of Computing and Data Science
Sigapore
gdeng003@e.ntu.edu.sg

Xingshuo Han
Nanyang Technological University
College of Computing and Data Science
Sigapore
xingshuo001@e.ntu.edu.sg

Guanlin Li
Nanyang Technological University
College of Computing and Data Science
Sigapore
guanlin001@e.ntu.edu.sg

Han Qiu
Tsinghua University
Institute for Network Sciences and Cyberspace
Beijing, China
qiuhan@tsinghua.edu.cn

Tianwei Zhang
Nanyang Technological University
College of Computing and Data Science
Sigapore
tianwei.zhang@ntu.edu.sg

ABSTRACT

Existing defense approaches against sensor spoofing attacks suffer from the limitations of limited specific attack types, requiring GPU computation, exhibiting considerable detection latency and struggling with the interpretability of corner cases. We developed PhyScout, a holistic sensor spoofing defense framework to overcome the above limitations. Our framework capitalizes on the observation that human drivers can rapidly and accurately identify spoofing attacks by performing spatio-temporal consistency checks of their environment. We commence by defining the generalized conflicts that different sensor spoofing attacks produce regarding the spatio-temporal consistency. These conflicts are subsequently unified and formalized through a least squares problem approach. This process is modeled using image-based feature point extraction and matching techniques, followed by the design of a risk identification method for each conflict.

We evaluate PhyScout across various environments, including simulators, datasets, and real-world scenarios. Compared to existing defense solutions, PhyScout offers rapid identification of sensor attacks (within 100ms) with low performance overhead (CPU-based), and conflict visualization. It demonstrates a fresh paradigm in autonomous vehicle security and presents new avenues for future research in robust and efficient defense mechanisms against sensor spoofing attacks. More video demos are at our anonymous website https://sites.google.com/view/physcout.

CCS CONCEPTS

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

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0636-3/24/10.
https://doi.org/10.1145/3658644.3670290

KEYWORDS

Autonomous Driving, Sensor Spoofing Attacks, Defenses

ACM Reference Format:
Yuan Xu, Gelei Deng, Xingshuo Han, Guanlin Li, Han Qiu, and Tianwei Zhang. 2024. PhyScout: Detecting Sensor Spoofing Attacks via Spatio-temporal Consistency. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670290

1 INTRODUCTION

Sensors, such as GPS, LiDAR, and cameras, serve as "eyes" for an autonomous vehicle (AV), interpreting real-world environmental information into structured data. This data is subsequently utilized by the autonomous driving system (ADS) to estimate vehicular states (e.g., pose) or environmental states (e.g., obstacles). However, these sensors also introduce potential vulnerabilities, providing an entry point for an adversary to hack into the vehicle system. Taking LiDAR as an example, an attacker can fool the object detection function by controlling several drones in specific positions to generate counterfeit laser points, causing the identification of nonexistent obstacles. Such manipulations can lead to forced braking on highways, potentially inciting rear-end collisions and compromising passenger safety [88]. Extensive research has been undertaken to investigate various categories of sensor spoofing attacks [56, 77, 79].

In response to the threat posed by sensor spoofing attacks, various defensive strategies have been proposed: (1) *Certified Defense* focuses primarily on detecting adversarial inputs and providing theoretical assurances of model robustness against white-box attacks. Common methodologies in this category encompass techniques such as objectness explaining [73], randomized cropping [35], interval bound propagation [82], derandomized smoothing [34], secure aggregation [72], feature space masking [74] and certified training [43]. (2) *Vision-based Consistency Checking* aims to incorporate additional sensor inputs into the deep learning model. The premise behind this strategy is to leverage the consistency across various sensor sources to identify potential attacks [42, 75].

Unfortunately, the above mitigation methods are encumbered by several notable constraints. (1) *Generalizability*: Current solutions are limited to addressing certain types of spoofing attacks on a specific sensor, such as object misclassification [34, 35, 42, 43, 72, 74, 75, 82] and obstacle hiding [73] on camera. However, it has been proven that almost all sensors could be potentially spoofed [77], necessitating multiple defensive strategies for comprehensive protection with a huge waste of onboard computing and storage resources. (2) *Detection Latency*: According to the industry standards published by Mobileye [57] and Udacity [2], the latency for processing tragic condition of ADS should be within 100 ms. However, current methods, particularly vision-based consistency checking [42, 75], require significant time (hundreds of milliseconds), which enhances the likelihood of vehicle collisions given the high speeds on highways [2, 57]. (3) *Interpretability*: Corner cases pose significant importance in ADS, yet current methods struggle to explain these instances owing to their reliance on machine learning methods. Recent researches [10, 64, 67, 85] highlight that these unexpected scenarios could lead to serious, even deadly, accidents. (4) *Performance Overhead*: Current checking methods involve either complex mathematical proofs or intricate deep learning processes to ensure rigorous robustness guarantees. The resultant high computational cost compromises their practicality for protection in autonomous driving as they need to compete with other critical applications for limited GPU resources.

This paper investigates the potential for a solution that addresses these constraints simultaneously, namely, ***a low-overhead, interpretable defensive method capable of real-time detection of sensor spoofing attacks***. We observe that human vision has the ability to detect sensor spoofing attacks in real time with robust interpretability. For example, teleportation or abrupt disappearance of obstacles would instinctively trigger a sense of incongruity. We attribute this perception to two factors: *spatial* and *temporal consistency*. Spatial consistency denotes that the estimated states of the vehicle should be consistent with the current spatial distribution. For instance, while a camera spoofing attack may deceive the victim vehicle into failing to detect forward obstacles, it does not physically eliminate the real objects. On the other hand, temporal consistency dictates that the trajectory of an object over continuous time should remain logically coherent and plausible. For example, an object cannot be intermittently recognized as a truck and then the sky. To translate this spatio-temporal consistency into a practical defensive method, it is imperative to address three issues:

- What are the conflicts present in existing spoofing attacks based on the principle of spatio-temporal consistency?
- How can we establish a methodology to formally express the spatio-temporal consistency, thereby enabling a precise quantification of the conflicts inherent in sensor spoofing attacks?
- How can these formalized conflicts be modeled and integrated within the existing ADS framework?

In response to the first question, we have identified seven attack goals inherent in existing sensor spoofing attacks: target position altering, target position deviating, target velocity destabilizing, obstacles appearing, obstacle hiding, object misclassification, and lane altering (Table 1). By analyzing them, we conclude four types of spatial consistency conflicts, namely global space tearing, global space shifting, target entity confusion, and target semantic confusion, along with a temporal consistency conflict, i.e., target flickering.

Secondly, by incorporating the vehicular state flow model, we formally interpret the aforementioned spatio-temporal consistency conflicts into a discrepancy between real observed measurements (those perceptible to the eyes) and predicted observed measurements (those expected cognitively). Subsequently, this discrepancy, considered as an error, is analyzed. The estimation of this error is then reformulated as the least squares problem for resolution.

Finally, to quantify this error, we introduce a method which makes predictions on feature point matching to detect sensor spoofing attacks. Initially, we extract 2D keypoints from each frame. Stable keypoints from successive frames are then triangulated to generate 3D map points. These map points are subsequently projected into the 2D space of the current scene, following the predicted estimated state alterations, and matched with real-time extracted 2D keypoints. The mean error across all matching pairs can be used to represent the spatio-temporal consistency. Accordingly, we design a specific detection scheme for each proposed conflict. All these methods are synthesized into one framework named `PhyScout`.

We perform comprehensive evaluations of `PhyScout` in both simulated and real-world environments. To replicate an authentic AV urban environment, we utilize Carla integrated with Apollo 7.0 as an ADS with the simulator. `PhyScout`, as a third-party module, processes the real-time frames, estimated pose, and detected bounding box to identify attacks. Experimental outcomes indicate that our method can detect all sensor spoofing attacks, drastically reducing the detection latency to less than 100ms compared to state-of-the-art defensive approaches. Moreover, our method can operate in real-time on one single CPU and visually present the error associated with each attack. Simultaneously, we conduct assessments with a physical vehicle to evaluate `PhyScout`'s practicality and adaptability in real-world conditions. The impact of various factors such as image resolutions, vehicle speeds, and environmental dynamics on our method is also estimated. The results confirm the persistent robustness of `PhyScout` under all assessed conditions.

2 MOTIVATION AND BACKGROUND

Sensor spoofing attacks have been proven to be quite effective in the ADS. However, they can be easily identified by humans by cognitive inconsistencies. This lack of coherence with reality arises from spatio-temporal conflicts between the spoofed target states and the actual states sensed in the real world. To elucidate the causes of these conflicts, we provide an overview of spoofing attacks and their background in § 2.1, as well as a discussion on the potential conflicts that may arise from each spoofing attack in § 2.2, which introduces our motivations to propose such a detection method.

2.1 Sensor Spoofing Attacks

In this paper, we focus on spoofing attacks on five mainstream sensors: GPS, LiDAR, camera, IMU, and ultrasonic sensor/MMW radar, which are fully or partially integrated into AVs for planning and control purposes. Spoofing of these sensors can cause the internal states estimated from sensor data to be modified, resulting in unexpected and dangerous actions such as rear-end collisions and falling off a cliff. To identify the sensor spoofing attacks and their

Table 1: Summary of existing sensor spoofing attacks and related spatio-temporal conflicts.

Attack Goal	Target States	Attack Vector	Spatial Consistency Conflict	Temporal Consistency Conflict
Target Position Altering	Position	GPS Spoofing [17, 23, 24]	Global Space Tearing	×
Target Position Deviating		GPS Spoofing [36, 54, 84]	Global Space Shifting	×
Target Velocity Destabilizing	Velocity	IMU Spoofing [47, 65, 66, 70]		×
Obstacle Appearing	Obstacle	Ultrasonic/MMW Radar Spoofing [44, 62, 80]	Target Entity Confusion	×
		LiDAR Spoofing [13, 27, 50, 58, 61], Camera Spoofing [48, 49, 69, 87]		×
Obstacle Hiding		LiDAR Spoofing [12, 27, 88], Camera Spoofing [15, 25, 37, 41, 71, 76, 86]		Target Flickering
Object Misclassification	Traffic Controller	Camera Spoofing [18, 19, 31, 41, 48, 49, 60, 68, 69, 81, 86]	Target Semantic Confusion	
Lane Altering	Lane	Camera Spoofing [28, 49, 55]		

target states, we summarize the findings from existing works and present them in Table 1. The table concludes seven attack goals associated with corresponding sensor spoofing attacks. Our proposed framework is designed to address these challenges and ensure the safety of AVs. In this context, spoofing attacks are classified based on the attack goals rather than the specific sensor. This approach ensures that our proposed defensive method is not confined to certain sensor characteristics, such as data formats and hardware prerequisites, thereby promoting broader applicability.

(1) Target Position Altering. This attack goal involves the adversary inducing the vehicle to locate at a predetermined position and trigger unexpected decisions and actions. A GPS spoofer is commonly employed to execute this attack, which generates false GPS signals that trick the AV's GPS receiver into providing inaccurate location, triggering the launch of some unexpected decisions and actions [17, 23, 24]. For example, the victim drone is forced to land immediately by spoofing a location to restricted no-fly zones [17].

(2) Target Position Deviating. This attack goal aims to accumulate position deviations and cause mis-predictions of a target vehicle's location. One of the ways to execute this attack is through the use of a GPS spoofer that continuously introduces small deviations to the AV's position [36, 54, 84]. For example, a lateral deviation could be employed to deceive the vehicle into moving away from the designated lane and colliding with the curb [36]. Another example involves slightly shifting the GPS location of an AV so that the spoofed navigation route aligns with the shape of the actual roads [84]. This approach can trigger physically plausible instructions that could compromise the safety of the vehicle and its occupants.

(3) Target Velocity Destabilizing. This attack goal is to generate an angular velocity in a single direction to cause the vehicle to overturn. AVs typically require dynamic adjustments to maintain balance in response to changes in velocity or position. These variances are estimated from IMU sensors, which an adversary can exploit by using an IMU spoofer to generate false data, thereby disrupting the balance [47, 65, 66, 70]. For instance, an attacker can destabilize a vehicle, causing the crash by spoofing fake IMU data [66].

(4) Obstacle Appearing. This attack goal aims to deceive a vehicle to mistakenly recognizing a non-existent obstacle in front. Typically, vehicles utilize various sensors such as LiDAR, camera, ultrasonic, and MMW radar to identify obstacles. However, it has been demonstrated that these sensors can be spoofed by either directly generating the false normal wave signals, such as ultrasonic [44, 80] and MMW radar [62], or by launching adversarial attacks on deep learning models through the use of projection [48, 49, 69, 87] or fake laser points [13, 27, 50, 58, 61]. For example, [48] demonstrated how an attacker can fool Tesla's autopilot into applying the brakes by projecting a phantom of a pedestrian on the road.

(5) Obstacle Hiding. This attack goal aims to deceive a vehicle into failing to detect an existing obstacle. Similar to the obstacle appearing attack, this attack can be implemented by spoofing LiDAR or cameras. Adversaries can use small patches or few laser points to deceive the internal deep learning model [12, 15, 25, 27, 37, 41, 71, 76, 86, 88]. For instance, by operating several drones in identified positions, adversaries can generate counterfeit laser points and interfere with the perception results of the point cloud model of the victim vehicle.

(6) Object Misclassification. This attack goal aims to cause a vehicle to misclassify traffic signs or traffic lights, resulting in hazardous consequences. Traffic controller detection systems rely on camera-captured images to perform classifications, making it possible for an attacker to deceive the deep learning model by projecting phantom images or attaching stickers [18, 19, 31, 41, 48, 49, 60, 68, 69, 81, 86]. For instance, recent research has focused on constructing robust adversarial examples to mislead real-world object detectors [86].

(7) Lane Altering. This attack goal aims to cause a vehicle to misidentify traffic lanes, leading to potentially dangerous consequences. Similar to the traffic controller detection module, the segmentation module also relies on camera-captured images to distinguish between different traffic lanes to take appropriate actions. Therefore, an adversary can deceive the segmentation module by projecting phantoms [28, 55] or attaching patches [49] to alter the AV's perception of the lanes. For example, previous research has explored the use of dirty road patches to deceive different lane detection models, causing the victim AV to deviate laterally [55].

2.2 Motivation

Sensor spoofing attacks have proven to be quite effective. However, their efficacy is limited by the ease with which they can be detected by human observers. Comprehending the underlying causality aids us in devising efficacious countermeasures against sensor spoofing attacks. This lack of coherence with reality arises due to conflicts between the spoofed target states and the actual states sensed in the real world. To describe these conflicts, we propose a definition of spatial-temporal consistency: (1) *Spatial consistency* refers to the variance of the spatial environment caused by the expected behavior of the system, which should be within an acceptable margin of error. (2) *Temporal consistency* pertains to the spoofer's ability to generate a continuous and steady stream of fake target states. By adhering to these principles, as Figure 1 shows, we identify 6 conflicts that arise between the goals of existing sensor spoofing attacks and the requirements of spatial-temporal consistency.

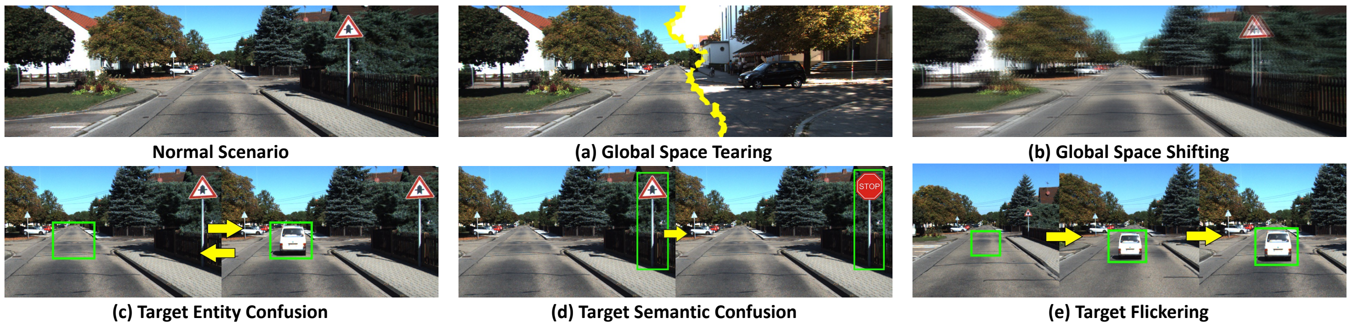

Normal Scenario
(a) Global Space Tearing
(b) Global Space Shifting
STOP
(c) Target Entity Confusion
(d) Target Semantic Confusion
(e) Target Flickering

Figure 1: Illustration of five spatial-temporal consistency conflicts.

(1) Global Space Tearing. (Figure 1.a) This spatial conflict can arise when an attacker spoofs the AV suddenly from one location to another, resulting in a phenomenon known as "space tearing". Due to the lack of physical continuity between these two locations, drivers can easily identify their vehicle's real position.

(2) Global Space Shifting. (Figure 1.b) The deviating position altering attack and the destabilizing velocity altering attack are two forms of attacks that aim to create a slight offset of a vehicle. This offset can cause a global space shifting conflict, which occurs when there is a mismatch between the expected and current sensing surroundings. Such conflicts can be easily identified by the driver, since the environment does not shift along with the spoofed direction.

(3) Target Entity Confusion. (Figure 1.c) This spatial conflict can arise when obstacles suddenly appear or disappear in the environment. While it is possible for the obstacle detection module to be compromised by spoofing attacks, the physical world remains unchanged. Thus, hiding targets can still be observed by drivers, and non-existent objects can be identified with a quick glance.

(4) Target Semantic Confusion. (Figure 1.d) This spatial conflict can arise in the environment when an object is misclassified by spoofers, leading to potentially dangerous actions. While it is possible for deep learning models in traffic controller detection modules to be compromised by spoofing attacks through minor changes to images, drivers can still identify anomalies through the texture and color of traffic signs and lanes.

(5) Target Flickering. (Figure 1.e) This spatial conflict can arise due to the robustness issues of adversarial spoofing attacks. Recent studies have revealed that these attacks demonstrate higher sensitivity to the target's distances and angles from the victim's sensors (e.g., camera and LiDAR) compared to normal scenarios [14, 37, 42, 83, 86]. This phenomenon can lead to the emergence of discrete time periods during which the spoofing attacks are effective, thereby providing the possibilities of detecting anomalies.

3 CONSISTENCY FORMALIZATION

This section aims to quantify the spatial-temporal consistency conflicts through the formulation of solvable mathematical expressions. This is accomplished by the introduction of a state flow model to represent the transition of AV states under both normal and attack scenarios (§ 3.1). By leveraging this model, we can derive formal representations of the conflicts. Subsequently, we tackle conflict identification by translating it into a nonlinear least squares problem (§ 3.2). This approach guides the design of our system.

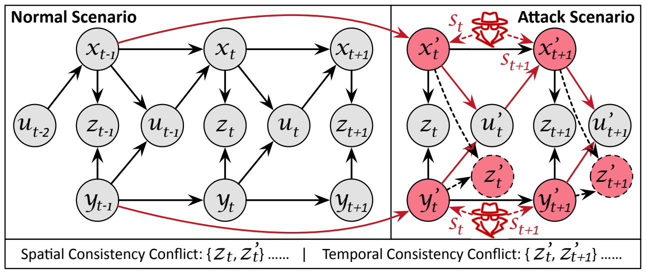


Figure 2: The state flow model of an AV system under the normal and spoofing attack scenarios.

3.1 AV State Model

An AV system can be modeled as a set of sensor inputs, system states, and control outputs, which change over time as the AV interacts with the environment. Meanwhile, the environment also changes in response to the AV's actions. In this paper, x_t and y_t denote the estimated pose and environment states based on sensor data at time t, whereas u_t is used to represent the control outputs that induce changes in x_t and y_t. Additionally, z_t signifies the direct observable physical environmental distribution under the pose x_t, i.e., the raw image captured by the camera. Figure 2 presents a state flow diagram of an AV system, where grey and red circles denote normal and spoofed states. The black and red solid lines indicate the direction of changes in normal and spoofed states.

Normal Scenario. The AV pose and environment states are subject to change as the AV moves. Specifically, the AV pose x_t at the current moment is determined by the AV pose x_{t-1} and the control action u_{t-1} at the last moment. The sensors capture these newly generated states and transform them into observations z_t. In addition, the estimated pose x_t and environment y_t are used to plan the control action u_t for the current moment. These processes of the state transition can be written as follows by using the motion model and measurement model:

$$\begin{cases} x_t = f(x_{t-1}, u_{t-1}) + w_t & \triangleright \text{Motion Model} \\ z_t = h(y_t, x_t) + v_t & \triangleright \text{Measurement Model} \end{cases} \tag{1}$$

where $f(\cdot)$ is a motion equation that describes the transition of a pose from x_{t-1} to x_t under the control output u_{t-1}, and $h(\cdot)$ is a measurement equation that describes the observation data z_t produced when the vehicle observes the environmental state y_t at x_t. v_t and w_t are the noises of these two state transitions and commonly assumed to satisfy the Gaussian distribution of the zero mean [63]:

$$w_t \sim \mathcal{N}(0, R_t), \quad v_t \sim \mathcal{N}(0, Q_t) \tag{2}$$

where $\mathcal{N}$ denotes the Multivariate Gaussian distribution, R_t and Q_t denote covariance matrices.

Attack Scenario. We consider the scenario where an adversary attempts to falsify the estimated states from $\{x_t, y_t\}$ to $\{x'_t, y'_t\}$ through sensor spoofing attacks s_t. This erroneous estimated data is then used by the AV to formulate the control action u'_t, which subsequently generates erroneous estimated state x_{t+1} and environment state y_{t+1}. The adversary can continuously spoof these states into $\{x'_{t+1}, y'_{t+1}\}$ through s_{t+1} and this error will continue to propagate until it reaches the attacker's desired end state x'_T, such as collision and rear-end. This process can be written as follows:

$$z'_t = h(y'_t, x'_t) + v_t \tag{3}$$

where the observation data z'_t is the state observed under the spoofed estimated states x'_t and y'_t. It is worth noting that z'_t cannot be observed since the control output u_{t-1} at the last moment is unchanged, the observation data is still z_t observed at x_t and y_t.

Spatial-Temporal Consistency Conflicts. It can be observed from Figure 2 that under the sensor spoofing attack, the transition of the system and environment state from $\{x_{t-1}, y_{t-1}; x_t, y_t; x_{t+1}, y_{t+1}\}$ to $\{x_{t-1}, y_{t-1}; x'_t, y'_t; x'_{t+1}, y'_{t+1}\}$. However, the former transition is continuous in the state space, while the latter is not. This is attributed to the misestimation of the states x'_t and y'_t generated by the spoofing attack, instead of being transferred from x_{t-1} and the output control u_{t-1} using the motion model. This paper focuses on the discontinuity of these estimated states and proposes *spatio-temporal consistency conflicts* as a means of characterizing this discontinuity. We can formulate these conflicts as follows:

$$\begin{aligned} z'_t \ominus z_t &> \varepsilon && \triangleright \text{Spatial Consistency Conflict} \\ z'_{t+1} \ominus z'_t &> \delta && \triangleright \text{Temporal Consistency Conflict} \end{aligned} \tag{4}$$

where $\ominus$ is an operation to quantify the difference between two environment states. ε and δ are two thresholds to check whether the difference is within a reasonable range.

- **Spatial Consistency Conflict.** This conflict is a result of the mismatch between the observation data z_t in the physical world and the predicted observation data z'_t based on the spoofed estimated states x'_t and y'_t.
- **Temporal Consistency Conflict.** This conflict is a result of the mismatch between the predicted observation data in different time periods, such as z'_t and z'_{t+1}.

3.2 Error Formalization

From the analysis of Equation 4, it can be inferred that the observation data z and the predicted data z' under an attack scenario exhibit certain discrepancies. This observation paves the way for conducting maximum a posterior estimation of spoofed pose x' and environment states y' while taking into account the known observation data z and control output u. We mark this maximum a posterior estimation as $(x', y')^*_{MAP}$ and formulate it as follows:

$$(x', y')^*_{MAP} = \underset{x', y'}{\arg\max}\, P(x', y' | z, u) \tag{5}$$

By invoking Bayes' rule, we can derive an estimation for the conditional distribution of the state variables as below:

$$\begin{aligned} (x', y')^*_{MAP} &= \arg\max \frac{P(z, u | x', y') P(x', y')}{P(z, u)} \\ &\propto \arg\max P(z, u | x', y') P(x', y') \end{aligned} \tag{6}$$

where $P(z, u|x', y')$ is the likehood and $P(x', y')$ is the prior. As the denominator of the conditional distribution does not depend on x' and y', it can be ignored. Moreover, since the prior is unknown, we can follow [20] to transform the maximum posterior estimation into a maximum likelihood estimation as follows:

$$(x', y')^*_{MLE} = \underset{x', y'}{\arg\max}\, P(z, u | x', y') \tag{7}$$

Equation 7 can be intuitively understood as identifying the state that is most probable to have generated the observation data. Obviously, there is a significant bias in the maximum likelihood estimates of the normal estimated state $(x, y)^*_{MLE}$ and the spoofed estimated state $(x', y')^*_{MLE}$.

By combining the measurement model in Equation 1 and the Gaussian distribution of noise in Equation 2, it can be inferred that the conditional probability in Equation 7 adheres to the Gaussian distribution as below [63]:

$$P(z_t | x'_t, y'_t) = \mathcal{N}(h(x'_t, y'_t), Q_t) \tag{8}$$

Considering a N-dimensional Gaussian distribution $x \sim \mathcal{N}(\mu, \sum)$, the expansion form of its probability density function can be expressed as follows:

$$P(x) = \frac{1}{\sqrt{(2\pi)^N \det(\sum)}} \exp(-\frac{1}{2}(x - \mu)^T \sum^{-1} (x - \mu)) \tag{9}$$

By applying the negative logarithm to both sides of the equation, we obtain the following:

$$-\ln P(x) = \frac{1}{2} \ln((2\pi)^N \det(\sum)) + \frac{1}{2}(x - \mu)^T \sum^{-1} (x - \mu) \tag{10}$$

As Equation 10 is a monotonically increasing function, maximizing the original function is equivalent to minimizing the negative logarithm. In the given formula, the first term is independent of x and can be omitted during minimization of x. Thus, as long as the quadratic term on the right is minimized, the maximum likelihood estimation of the state can be obtained. By combining with Equation 7 and 8, the maximum likelihood estimation can be expressed as follows:

$$\begin{aligned} (x', y')^*_{MLE} &= \arg\max \mathcal{N}(h(x'_t, y'_t), Q_t) \\ &= \arg\min e^T_{z,t} Q_t^{-1} e_{z,t} \\ \text{w.r.t.} \quad e_{z,t} &= z_t - h(x'_t, y'_t) \end{aligned} \tag{11}$$

This equation can be considered a method for minimizing a quadratic form of the noise variable, commonly referred to as the *Mahalanobis distance*. In this context, the Mahalanobis distance represents the weighted Euclidean distance (two-norm) by the information matrix Q_t^{-1}. The information matrix Q_t^{-1}, also known as the inverse of the covariance matrix in the Gaussian distribution, plays a crucial role in this equation.

Temporal consistency is a critical aspect of our analysis, whereby we evaluate data at batch times. In addition, our assessment of observation data and environment state takes into account various

Table 2: Definition of stealthiness and naturalness.

Constraints	Constraints	Examples
Stealthiness	Active Spoofer	Signal projector
	Remote Attack Range	$> 10m$
	Imperceptible Signal	GPS, laser, ultrasonic, MMW Radar
Naturalness	Passive Spoofer	Patches, objects
	Semantic Independence	Target semantics cannot be modified

features, such as pixels or corners, that coexist at the same moment. We use $z_{t,k}$ and $y_{t,k}$ to denote each future of observation data and environment states at time t. Typically, these control outputs and features can be assumed to be independent of one another at each moment. As a result, we can factorize the conditional probability:

$$P(z,u|x',y') = \prod_t P(u_t|x_{t-1},x_t) \prod_{t,k} P(z_{t,k}|x'_t,y'_{t,k}) \tag{12}$$

We can minimize the Mahalanobis distance between the estimated values at all times and the real observation data, which is equivalent to seeking the maximum likelihood estimation. By leveraging negative logarithms in Equation 10, we can convert products in Equation 12 into sums as bellows.

$$\begin{aligned} &\min J(x',y') = \sum_t e_{u,t}^T R_t^{-1} e_{u,t} + \sum_t \sum_k e_{z,t,k}^T Q_{t,k}^{-1} e_{z,t,k} \\ &\text{w.r.t.} \quad e_{u,t} = x'_t - f(x'_{t-1}, u_{t-1}) \end{aligned} \tag{13}$$

where minimizing $J(x',y')$ can be interpreted as a least squares optimization problem, which is synonymous with maximizing the likelihood estimation of states x' and y'. Refining any estimation is capable of marginally reducing the error, usually approaching a minimum. Hence, the stated process is a standard instance of non-linear optimization. This paper employs the *Levenberg-Marquardt* algorithm to address the optimization quandary. It is worth noting that all assumptions in this section align with prevailing standards in the robotics domain [59, 63], are corroborated by extensive empirical testing, and have been incorporated into current ADS.

4 SYSTEM AND THREAT MODEL

System Model. We consider an end-to-end AV application that utilizes multiple sensors including GPS, IMU, camera, LiDAR, ultrasonic, and MMW radar. The GPS and IMU sensors are employed for vehicle localization in the map, which is then used to plan the optimal route toward the destination. To ensure safety during navigation, the application uses camera, LiDAR, ultrasonic, and MMW radar sensors to estimate the distance between the vehicle and obstacles. When the obstacles are too close, the vehicle will decelerate or brake to avoid a collision. Additionally, the camera sensor is utilized to recognize regions of interest (ROI), such as traffic lights, traffic signs, and lanes. The detection of these traffic controllers triggers various events for safe decision-making and control. Our method is a universal detection framework that is independent of the specific model used by the perception module. The state estimation can stem from a single sensor or multiple sensors' fusion [12, 21, 36, 52], which makes our application more practical.

Threat Model. We consider sensor spoofing attacks aiming at achieving one of seven attack goals outlined in Table 1. These attacks are designed to cause the vehicle to make dangerous decisions and take action, such as colliding with an obstacle or stopping on a highway. We assume that adversaries could create fictitious obstacles or conceal real ones, or alter the interpretation of traffic controllers and lanes. However, such capabilities are limited with the *stealthiness and naturalness constraints*: As detailed in Table 2, stealthiness is characterized by the use of an active spoofer, which projects malicious signals from a distance. Such an attack remains inconspicuous to drivers even at significant ranges, such as beyond 10 meters, where the attacker can be easily hidden in such a remote distance. Moreover, the imperceptible features of the emitted signals, such as GPS, laser, ultrasonic, and millimeter-wave radar, cannot be observed to the naked eye, thereby preserving the covert aspect of the attack. On the other hand, naturalness is illustrated through passive spoofing methods, where the spoofer is either affixed to or placed in proximity to the intended target. This form of attack is well-designed so that it does not alter the target's inherent semantics, ensuring the alteration remains undetectable to the driver. As such, while natural elements like dirt patches on roads [55] and small-scale stickers on traffic signs [26, 77] are permissible within our model, overt actions such as covering the target entirely with an untextured sticker or adding distinct semantic symbols are excluded from our work.

Our model excludes large-scale or multi-point alterations. This assumption was predicated on the belief that alterations affecting multiple points within a camera's field of view might inherently diminish the attack's stealthiness due to the introduction of more discernible anomalies. Although [13] has shown the feasibility of the multi-point LiDAR spoofing attack, we recognize that this may not translate directly to camera systems, given their distinct detection capabilities and vulnerabilities. In addition, we also acknowledge that the use of a single modality, i.e., camera for detecting sensor spoofing attacks has its limitations, which would cause a single-point fault. Our model does not consider scenarios with extremely low visibility due to very dim light or heavy fog, because these scenarios are also an open problem and challenge for commercial autonomous driving systems and have not been completely solved. We do not consider cyber attacks against RVs, such as those targeting software and ROS vulnerabilities [51, 78], in-vehicle networks [8, 9, 32], DNN backdoors [22], communication protocols [16, 29], and side-channel leakage [40], as they do not directly target on-vehicle sensors.

5 METHODOLOGY

This section presents our methodologies for detecting sensor spoofing attacks. We begin by outlining some of the technical challenges of designing a universal detection framework(5.1). Next, we introduce our proposed solutions to address these challenges (5.2-5.3).

5.1 Technical Challenges

(1) Modeling consistency conflicts. As humans are able to identify anomalies in spatial-temporal conflicts, we need to emulate this ability by modeling these conflicts. Meanwhile, we need to ensure the *robustness* of our model against spoofing attacks, ensuring that it can accurately identify any sensor spoofing attacks.

(2) Detection Scheme Design. To effectively mitigate various sensor spoofing attacks, we need to design different detection schemes

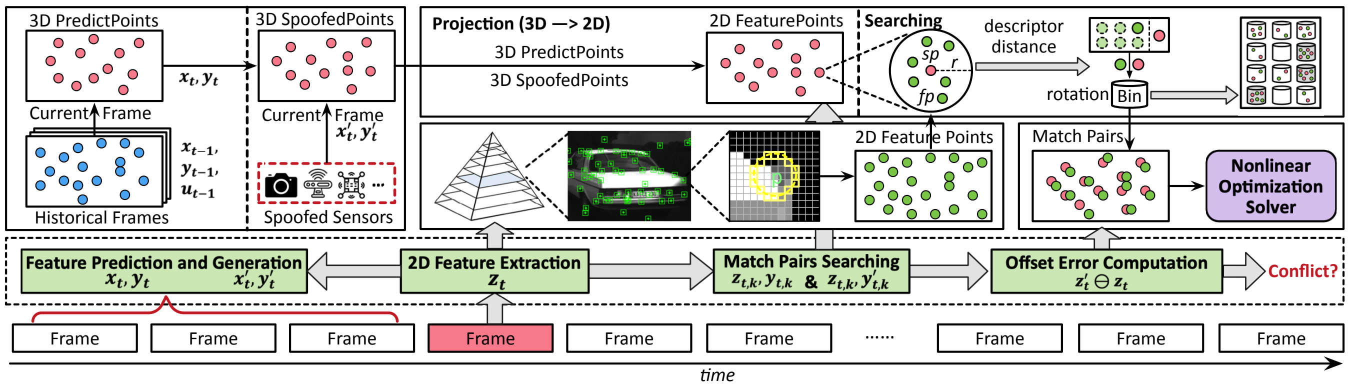


Figure 3: The spatial consistency model.

based on our proposed model. Additionally, to ensure the *completeness*, these schemes must be designed based on a thorough understanding of the various types of potential attacks.

(3) Optimizing performance. In the context of physical AVs with limited computing resources, ensuring the *real-time* execution of a detection framework is critical for mitigating risks. Thus, we need to propose a performance optimization approach that enhances the efficiency of our detection methodologies, allowing the vehicle to promptly identify spoofing attacks and mitigate associated risks.

5.2 Consistency Conflicts Modeling

5.2.1 Spacial Consistency Conflicts Modeling. We investigate the spatial consistency by checking the variance of the spatial environment caused by its expected behavior. To model this variance, we propose extracting features from sensor data to represent the distribution of the surroundings. It is crucial for these features to be robust enough to be extracted at any continuous space. By comparing the error between the extracted features z_t and the predicted features z'_t, we can establish a measure for the conflict of the spatial consistency. Additionally, these extracted features should be capable of identifying target semantics, enabling us to differentiate between different traffic controllers. Figure 3 shows our spatial consistency model. It consists of four components as follows.

2D Feature Extraction. This component aims to extract 2D feature points from the current perceived image, i.e., measurement data z_t. It is based on the Oriented FAST and Rotated BRIEF (ORB) algorithm [53]. We choose ORB rather than other popular features such as Scale-Invariant Feature Transform (SIFT [39]), Speeded Up Robust Features (SURF [6]), and Accelerated-KAZE (A-KAZE [4]) due to its real-time performance without requiring GPUs and its good invariance to changes in viewpoint and illumination [45]. The 2D feature extraction process begins by selecting a pixel p as the center and searching for 16 pixels on a circular pattern with a radius of 3 pixels. If the difference in brightness between the searched pixels and p exceeds a predetermined threshold (e.g., 20%) for 12 consecutive pixels, p is considered as a keypoint and selected as a feature. These keypoints are typically located in corners, edges, and blocks within the images. To ensure scale and rotation invariance in the vehicle's movement, we further process the extracted keypoints using multi-scale image pyramids and intensity centroid algorithms. An image pyramid represents a single image at multiple scales, with each level being a downsampled version of the original image. All keypoints extracted from various levels are then collected. The intensity centroid algorithm [53] leverages the fact that a corner's intensity is the offset from its center to determine the orientation. Finally, to differentiate the keypoints further, we use a descriptor, which is a 256-bit binary feature vector consisting of only 1s and 0s. Each bit is computed by comparing the brightness of surrounding pixels based on a predefined pattern.

Feature Prediction and Generation. This component aims to predict 3D feature points y_t by utilizing the historical environment states $[y_0, \ldots, y_{t-1}]$ and generating counterfeit 3D feature points y'_t from spoofed sensor data. To estimate the normal state x_t and y_t, we incorporate mapping mechanism from the vision-based SLAM [46], which converts stable keypoints from the current frame into map points by triangulating the current frame with other adjacent frame and stores all historical map points into a map dataset. A keypoint is deemed stable if it satisfies two criteria: the number of frames observing the map point is more than two since its creation, and the ratio of the number of frames that can track the map point to the number of frames that can observe this map point is greater than 25%. We define that if the current frame can find a matching key point for the map point, it is trackable, and if only the map point can be observed, it is observable. In addition, we assume that the objects in adjacent frames move uniformly in a short period of time, thus the pose of the current frame x_t can be estimated based on the pose of the previous frame x_{t-1} and related control outputs u_{t-1}. The environment state of the current frame y_t can also be predicted in the same method. Note that this hypothesis is also applicable in mainstream SLAM frameworks [7, 11, 46] and has shown strong robustness in our subsequent experiments. To determine the spoofed state x'_t and y'_t, we combine the keypoints extracted in the current frame with perceived data from spoofed sensors and formulate corresponding strategies for different spatial consistency conflicts, which are elaborated in § 5.3. It is worth noting that the estimation of x_t and y_t is based on the distribution of keypoints in captured images and is not affected by spoofed data.

The generation of map points for each frame occupies substantial computational and storage resources. To mitigate this performance overhead, we align with the ORB-SLAM framework, where the conversion of stable observed keypoints to map points for a frame is performed only only when the current frame is identified as a keyframe. The determination of keyframes is typically based on whether the environmental changes surrounding the vehicle are substantial. This approach significantly mitigates the frequency of calculations required. Moreover, our method has no need for constructing an environment map, as a window mechanism is established wherein only map points generated by N continuous keyframes are stored. Based on this way, the growth in storage requirements due to map expansion is considerably reduced.

Match Paris Searching. This component aims to identify keypoint matching pairs from the 2D extracted keypoints z_t and 3D predicted points y_t or spoofed points y'_t. To do this, we need to first project 3D map points in the physical world to the 2D coordinate in the current frame with an estimated pose, which is formalized as:

$$(m_i^x, m_i^y, 1)' = I_C \cdot (\frac{M_i^x}{M_i^z}, \frac{M_i^y}{M_i^z}, 1)'$$
$$\text{w.r.t.} \quad I_C = \begin{bmatrix} f_x & 0 & c_x \\ 0' & f_y & c_y \\ 0' & 0 & 1 \end{bmatrix}, \quad \begin{aligned} m &= (m_1, m_2, \dots, m_{n_m}) \\ m_i &= (m_i^x, m_i^y), \end{aligned} \tag{14}$$

Here (M_i^x, M_i^y, M_i^z) and (m_i^x, m_i^y) are the 3D map point and its projected coordinates in the current frame. I_C is the camera intrinsic matrix containing the focal length along x-axis f_x and y-axis f_y, and the optical center offset along x-axis c_x and y-axis c_y.

Then we need to identify the corresponding keypoint k_i in the current frame for each projected map point m_i. Specifically, we traverse all map points and create a container K_i for each map point, storing all keypoints within a circular area of radius r from the map point. As we mentioned in § 5.2.1, each extracted keypoint k is allocated to one level l_k of multi-scale image pyramids. Since M is transferred from keypoints, each m also owns a level l_m. The keypoint matching follows a rule that level l_{k_i} of the selected keypoints k_i must be the same as level l_{m_i} of the corresponding map point m_i or the difference is 1, which is formalized as:

$$K_i = \{k_j = (k_j^x, k_j^y) | (m_i^x - k_j^x)^2 + (m_i^y - k_j^y)^2 \le r^2\}$$
$$\text{s.t.} \quad l_{k_j} \in \{l_{m_i} - 1, l_{m_i}, l_{m_i} + 1\}, \text{for each } \ k_j \in K_i \tag{15}$$

For each keypoint in the container, the feature searching function computes the Hamming distance of descriptors between all keypoints $k_j \in K_i$ and the related map point m_i. The keypoint with the shortest distance $k^{\text{best}} \in K_i$ will form a pair with the map point m_i and be put into the corresponding histogram bin according to the rotation angle difference between them. Histogram bins are twelve containers for storing matching pairs of different angular differences (interval: 30°). Finally, the three bins with the most matching pairs (m_i, k^{best}) are selected as inputs to the model, while others are discarded. We can formulate this process as following:

$$(m_i, k^{\text{best}}) \in \{(m_i, k^{\text{best}}) | \text{BinIdx}((m_i, k^{\text{best}})) \in \text{Top3BinIdx}\}$$
$$\text{w.r.t.} \quad k^{\text{best}} = \min \text{HDist}(k_j, m_i), k_j \in K_i$$
$$\text{BinIdx}(m_i, k^{\text{best}}) = \frac{|\text{angle}_m - \text{angle}_{k^{\text{best}}}|}{30} \tag{16}$$

Here HDist and BinIdx denote the functions of computing the Hamming distance and the bin index. Top3BinIdx is the set of top three bin indexes.

Offset Error Computation. After the aforementioned procedures, we have acquired all the necessary components for solving Equation 13, including the observed data z_t, the normal predicted pose x_t, and the environment state y_t, as well as the deceived pose x'_t and environment state y'_t. In order to perform the optimizations, we utilize the graph nonlinear optimizer g2o [33], which is based on the *Levenberg-Marquardt* implementation. Our primary objective is to minimize all error edges by traversing an error edge for each matching pair, with the intention of discovering an appropriate pose x_t and environment state y_t. Following multiple optimizations, we will calculate the total mean of the error and identify the matching pair whose error exceeds the threshold as an outlier.

5.2.2 Temporal Consistency Conflicts Modeling. We investigate the temporal consistency of a system by analyzing the variance in the bounding boxes generated by object perception modules in ADS. To model this variance, we measure the error between the extracted features z_t^B and the predicted features $z_t'^B$ within the bounding box. Since $z_t'^B$ is derived from $z_{t-1}'^B$, the error between z_t^B and $z_t'^B$ can be employed to verify temporal consistency. If an obstacle abruptly disappears or transforms into another object, the distribution of feature points should correspondingly shift, resulting in a substantial error between time $t-1$ and t. This discrepancy affords us the means to identify such attacks by examining the error and the variance of the bounding boxes.

5.3 Detection Scheme Design

In this subsection, we present various detection strategies targeting distinct spatial consistency conflicts, as summarized in Table 1. These strategies share a common goal, which is to generate the current false environment state, i.e., y'_t, using the spoofed sensor data in conjunction with the known system state x_{t-1}, y_{t-1}, and u_{t-1}. Figure 4 provides an overview of the workflow for each detection scheme, which will be elaborated upon in detail below:

Normal Scenario. In the normal scenario, the current environment state y_t can be inferred from the previous environment state y_{t-1} and the system control output u_{t-1}, utilizing the equation:

$$[y_t, 1]' = T \cdot [y_{t-1}, 1]'$$
$$\text{w.r.t.} \quad T = \begin{bmatrix} R & r \\ 0' & 1 \end{bmatrix}, \quad y_i = (y_i^x, y_i^y, y_i^z), \tag{17}$$

where y_t and y_{t-1} denote the position of 3D map points at time t and $t-1$. T is a 4x4 transformation matrix to represent the transformation relationship from y_{t-1} to y_t and can be calculated from u_{t-1}. Specifically, the transformation matrix is a diagonal matrix containing a 3x3 rotation matrix R and a 3x1 translation matrix r. The immunity of Equation 17 to sensor spoofing attacks is attributed to its ability to function independently of sensor data other than images.

Global Space Tearing. The global space tearing scenario is characterized by a significant deviation between the false position and the actual position of the vehicle. However, the offset error of the feature points from the previous and current frames is found to be negligibly small. Thus, we introduce the identification conditions that must be met for this attack to occur: $(pos_t - pos_{t-1} > \theta)$ && $(z'_t \ominus z_t < \varepsilon)$. Here pos and θ denotes the position of the AV and the threshold to check whether the difference between previous and current positions is within a reasonable range.

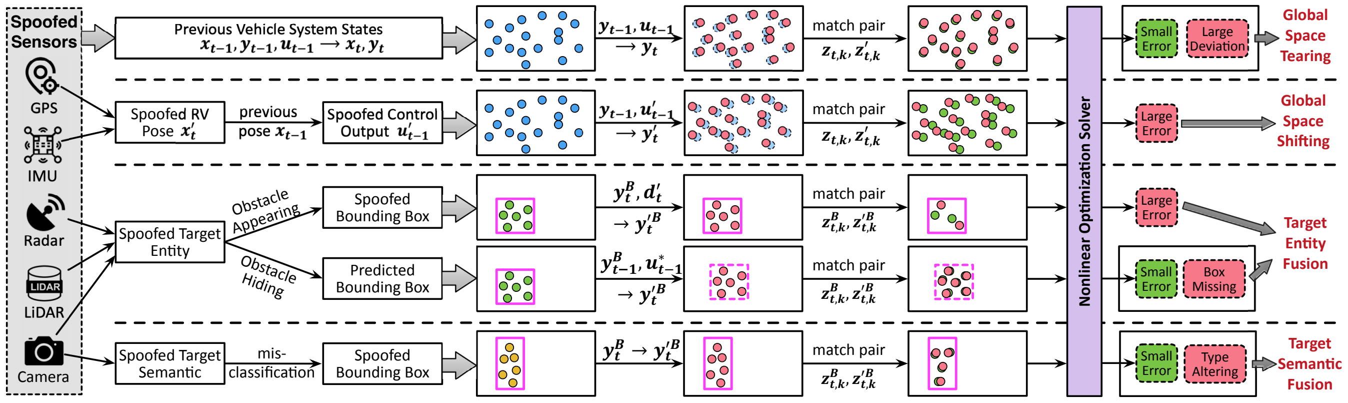


Figure 4: The detection schemes for each type of spoofing attacks.

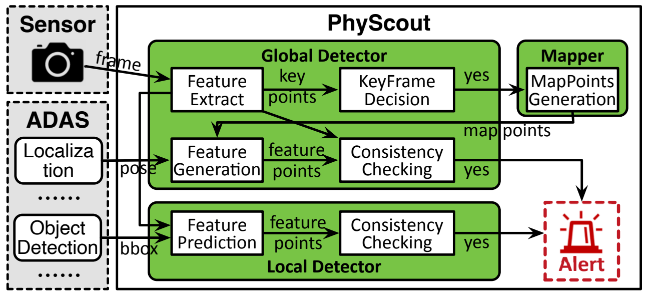


Figure 5: The overview of the Phycout.

Global Space Shifting. In the context of global space shifting, the position of the vehicle will undergo slight shifts, but the resulting changes in direction will be significantly different. In order to address this issue, we propose a method to obtain the current control output by analyzing the spoofed IMU or GPS data and comparing it with the previous pose x_{t-1}. We note that the resulting error u'_{t-1} is distinct from the control u_{t-1} output by the system at the previous moment. This discrepancy leads to an incorrect transformation matrix T' and we can calculate the spoofed environment state y'_t using Equation 17. These 3D map points are then mistakenly projected to 2D estimated points z'_t using Equation 14 and matched with corresponding keypoints z_t using Equation 15 and 16. Obviously, the actual observed state z_t and z'_t would exhibit significant errors during nonlinear optimization: $(z'_t \ominus z_t > \varepsilon)$.

Target Entity Fusion. In the context of the target entity confusion scenario, the vehicle will either fail to detect an obstacle or mistakenly identify a non-existent obstacle ahead. In order to address the obstacle appearing attack, we propose a methodology that involves the extraction of all map points y^B_t within the bounding box. These 3D points are then converted into deceived environmental features y'^B_t by attaching the distance d'_t estimated by the objection detection module. These false depth data could project y'^B_t into z'^B_t with incorrect level. Due to the different distance between the real observation z^B_t and spoofed estimated feature z'^B_t, the error during nonlinear optimization will increase substantially: $(z'^B_t \ominus z^B_t > \varepsilon)$.

To mitigate the obstacle hiding attack, we need to map the map points within the bounding box $bbox_t$ that were detected in the previous time y^B_{t-1} to the current time y'^B_t. However, the bounding box cannot be detected during the attack, making it impossible to use the above approach. Thus, we infer the transformation matrix T from the bounding box in the previous moment to the current time based on the relative speed u^*_{t-1} of the target and the vehicle. This data can help us to predict the position of the bounding box in the next short period of time. The mapped environment states y'^B_t are then projected into z'^B_t and nonlinearly optimized with their corresponding matching observation keypoints z_t, and the error value is expected to be as small as in the normal scenario. Thus, the identification conditions that must be met for this attack to occur: $bbox_t \to NULL$ && $(z'^B_t \ominus z^B_t < \varepsilon)$.

Target Semantic Fusion. In the scenario of target semantic confusion, the vehicle might misclassify objects, erroneously identifying one object or lane as another. To address this issue, we propose an approach similar to the method used for detecting obstacle appearing attacks. However, rather than attaching the obstacle distance d to the current map point, we continue utilizing the distance derived from triangulating the original keypoints. Once the type of the detected object changes, the distribution of its feature points is expected to shift correspondingly. Consequently, a substantial discrepancy between the observed value z^B_t and the predicted observed value z'^B_t should arise. This discrepancy can be used to identify such the attack: $type_t(A \to B)$ && $(z'^B_t \ominus z^B_t < \varepsilon)$. Note that our detection of both obstacle hiding attack and misclassification attack also leverage the phenomenon of target flickering conflict.

6 PHYSCOUT

In this paper, we present Phycout, a novel framework that has been developed to effectively detect sensor spoofing attacks using the above-mentioned spatio-temporal consistency detection schemes. Figure 5 shows an overview of Phycout, which mainly consists of three working threads: *global detector*, *local detector* and *mapper*.

Global Detector. This module identifies sensor spoofing attacks from global spatial disruptions and shifts. It begins by extracting 2D keypoints from each frame. These features are then utilized to generate 3D map points and construct optimization matching pairs. As discussed in § 5.2, only when the frame is identified as a keyframe are the keypoints transferred to the subsequent module. Historical map points are transported to the current moment and projected into 2D space. These projected map points seek out the best features in the extracted keypoints to form an optimal matching pair and compute the error. Ultimately, these errors in conjunction with pose data serve to identify global spoofing attacks.

Table 3: Implementation of each spatial-temporal conflict.

Conflict	Case Attack Goal	Target Node	Target Channel	Malicious Data	Duration (Frames)	Threshold
Global Space Tearing	Target Position Altering	rtk_localization	/localization/pose	(1, y, z, qx, qy, qz, qw)	10	20
Global Space Shifting	Target Position Deviating	rtk_localization	/localization/pose	(x, y+0.2, z, qx, qy, qz, qw)	10	40
Target Entity Confusion	Obstacle Appearing	v2x_fusion	/apollo/perception/obstacles	Yolo4BBoxes(x, y, z,...)	100	10
Target Entity Confusion Target Flickering	Obstacle Hiding	v2x_fusion	/apollo/perception/obstacles	Yolo4BBoxes(999, 999, z, ...)	50	10
Target Semantic Confusion Target Flickering	Object Misclassification	v2x_fusion	/apollo/perception/obstacles	Yolo4BBoxes(object class: bird, ...)	50	10

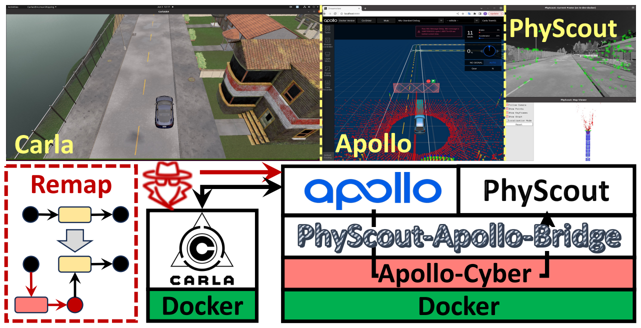


Figure 6: The experimental setup of simulated environment.

Mapper. This module generates global and local 3D map points from historical 2D keypoints. Global map points are constructed from keypoints extracted from two adjacent keyframes, while local map points are generated from keypoints extracted from the bounding boxes of two successive frames. We use frames instead of keyframes to generate local map points because the sparsity of keyframe-generated map points in the local space could cause significant error fluctuations when detecting local attacks.

Local Detector. This module identifies sensor spoofing attacks stemming from target entity and semantic confusion. Much like the global detector, it projects the historical 3D map points within the bounding box to the 2D predicted points at the current moment. These features are then used to seek out the best matches within the extracted keypoints in the bounding box to form an optimal pair and calculate the error. Ultimately, these errors and the bounding box, serve to identify local spoofing attacks.

7 DEFENSE ANALYSIS AND EVALUATION

7.1 Experimental Environment

Setup. In our evaluation, we employ PhyScout as an evaluation tool, integrating it with the cutting-edge open-source ADS, Apollo 7.0, and the Carla simulator. As depicted in Figure 6, separate docker environments are designated for Apollo and Carla. Specifically, Apollo derives environmental and vehicular contexts from Carla, enabling the planning of optimal routes to destinations and the calculation of instant velocities. Subsequent control commands are relayed to Carla to propel the AV. PhyScout is co-located in the same docker as Apollo, designed to receive sensor data (e.g., image and pose) through an established bridge for the detection of potential attacks. Notably, the bridge utilizes the Apollo Cyber module, serving as a pivotal mechanism for data conversion. All frameworks are deployed in a powerful server (i9-10900 CPU@2.8GHz with 32GB of RAM and NVIDIA GeForce RTX 2080Ti) in our lab.

Attacks. We assume that the attacker possesses robust adversarial capabilities, sufficient to reliably launch an attack. Therefore, we directly manipulate the internal estimation state within the AV. This is achieved through a systematic identification of the target channel, along with its associated readers and writers within the Apollo system. The red dotted box in Figure 6 depicts our approach, where we remap each writer's subscribed channel to a new channel, subsequently deploying a malicious node that transforms messages from the new channel to the target channel. The decision to directly modify sensor data in our simulation experiments is predicated on two fundamental advantages: (1) this approach allows us to simulate sensor spoofing attacks that are more consistent and stable than those typically possible in physical settings, representing a more consistent and stealthier threat model that is challenging to detect. Evaluating our system, PhyScout, against such rigorous conditions is crucial for assessing its robustness. (2) it affords us flexibility to instantiate a diverse array of attacks across varying times and locations, thereby subjecting PhyScout to a rigorous examination across multiple environmental contexts.

However, we also recognize that discrepancies persist between simulated sensor input modifications and their real-world implementation. This gap is primarily due to the different influences that physical adversarial objects can have on camera-based spoofing attacks, such as the variation in extracted keypoints and corresponding map points. Furthermore, the feasibility of our proposed metrics in physical settings need to be further validated.

Table 3 comprehensively details the configurations of the target channel, along with the malicious data for different conflicts. Since each sensor spoofing attack goal listed in Table 1 is aligned with at least one of our proposed five spatio-temporal conflicts, our evaluation of PhyScout's detection on sensor spoofing attacks is conducted from these five distinct conflicts. This approach allows us to more thoroughly evaluate PhyScout's effectiveness. PhyScout is capable of identifying even new types of attacks, as long as they violate spatio-temporal consistency. The malicious data, propagated by the malicious node, is dispatched to every subscribed channel associated with each reader. For each spatial-temporal conflict, we implement a case attack goal to evaluate our defensive methods.

Threshold Selection. Within the PhyScout, heuristic thresholds can be classified into two primary groups: (1) thresholds for extracting feature points and generating map points, and (2) thresholds for detecting sensor spoofing attacks. For the first group, our module mechanism leverages the well-established SLAM framework technology [7, 11, 46]. The thresholds adopted are influenced by these frameworks, which have demonstrated real-world robustness. So they are well-studied and reliable. For the second group, our thresholds were determined through extensive experiments. Each attack was tested in five unique scenarios with different keypoint distributions. Both normal and attacked scenarios were repeated five times to observe error variations. The significant difference

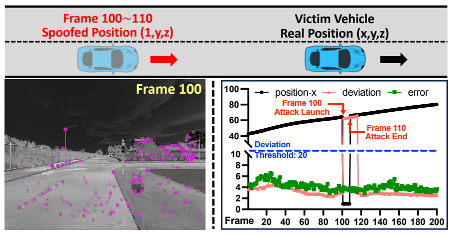


Figure 7: Global space tearing attack.

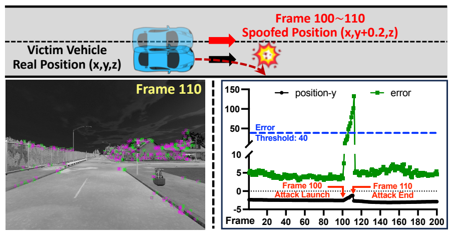


Figure 8: Global space shifting attack.

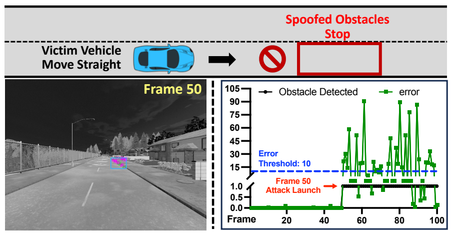


Figure 9: Obstacle appearing attack.

between the maximum threshold in normal conditions and the minimum in attacked scenarios led us to empirically choose the current thresholds, ensuring coverage of all attack scenarios.

7.2 Attack Detection

Global Space Tearing. To implement the target position altering attack, we experimentally adjusted the coordinate parameters of an AV within a global coordinate system, generating the illusion of the AV's teleportation to a distinct location. As illustrated in Figure 7, starting from the 100th frame, the AV's x-coordinate shifts from its initial value to 1 (highlighted in red) over a span of ten frames, preserving the y, z coordinates and the direction unchanged. Post-attack, the original x-coordinate is reinstated.

As derived from the bottom-right figure, we notice that the x-coordinate's decrease from 60 to 1 (black circles) at the 100th frame correlates with a surge in pose deviation (red triangles) from 4 to 60. This exceeds the predefined threshold of 20 (dashed blue line). Given our pose deviation calculation maintains a 10-frame window, an anomaly spanning 20 frames is detected. It is noteworthy that the optimization error (green squares) remains within normal bounds during the attack, since the spatial continuity of the physical environment observed by the AV is retained. The bottom-left image provides a representation of the 2D map point projection's location (purple squares) at the 100th frame, which closely aligns with the currently observed 2D keypoint (green squares).

Global Space Shifting. To implement the target position deviating attack, we applied an incremental offset (+0.2) to the AV's y-coordinate on a frame-by-frame basis after 100th frames, while keeping the x and z coordinates unaltered. As depicted in Figure 8, due to the miscalculation of its position, the AV erroneously presumes it has deviated from its lane, causing a right turn to regain the supposed lane (red dotted line). Consequently, the AV, while in the correct position, collides with a roadside obstacle. Post-attack, the original y-coordinate is restored at frame 110.

The bottom-right image reveals that as the y-coordinate (black circles) gradually ascends from -3 to -1 at frame 100, the optimization error (green triangles) experiences a substantial surge to 132. This discrepancy arises due to the mismatch between the space under the AV's estimated pose and the observed space under the ground truth pose. Notably, as this value significantly surpasses our predefined threshold of 40, the attack can be readily detected. Further, the lower-left figure highlights an apparent offset error at frame 110 between the projected 2D map point (purple squares) and the contemporaneously observed 2D keypoint (green squares).

Obstacle Appearing. To implement the obstacle appearing attack, we inject malicious data (x, y, z) to the obstacle-related topic, causing the misestimation of the obstacle-sensitive fusion node, termed v2x_fusion. Our methodology involves the generation of a specified bounding box, strategically placed in the front of the targeted vehicle, to simulate a nonexistent obstacle. As Figure 9 shows, the victim vehicle will suddenly decelerate and engage braking mechanisms to avoid collision with the phantom obstacle. We use a blue square to represent our malicious bounding box. Within this box, the purple and green squares are the reprojected 2D extracted keypoints and matched 2D original keypoints base on reprojected keypoints.

The lower right section of Figure 9 presents a fluctuation analysis of the error between reprojected 2D keypoints and their corresponding 2D keypoints as extracted from a continuous sequence of 100 frames. Notably, the obstacle appearing attack was launched at the 50th frame, leading to the abrupt detection of a phantom obstacle by the victim vehicle (as indicated by the value of the black line equaling 1), despite its nonexistence evident from the 50th frame's simulator snapshot. This misleading detection results from the incorporation of inaccurately estimated bounding box distances into the keypoints extracted from distant curbs, thereby leading to erroneous mapping during the matching process, and causing the creation of an obvious discrepancy between the matched keypoints and their original counterparts. Consequently, the error value witnesses an exponential surge at the 50th frame, surpassing the pre-set threshold of 10 for obstacle attacks. This obvious increment allows for the effortless detection of the attack.

Obstacle Hiding. To implement the obstacle hiding attack, we inject malicious data (999, 999, z) to the obstacle-related topic. Our attack aim to replace the target vehicle in front to another position, the bounding box that should have been detected from the target will be disappeared. As Figure 10 shows, the bounding box depicted as a blue box is not estimated from the target, but the predicted box to extract keypoints for the attack detection. The purple and green squares inside the boxes denote the reprojected 2D extracted keypoints and the 2D original keypoints matched against these reprojected keypoints. Building on this, the victim vehicle will erroneously presume the absence of forward obstacles, resulting in a collision with the vehicle ahead, as opposed to a preventive stop.

Similar to the obstacle appearance attack, we record the variation of the error value fluctuations over a sequence of 100 continuous frames. The obstacle hiding attack is launched at 50th frame, as depicted in the lower right segment of Figure 10, leading to the victim vehicle's sudden inability to detect obstacles (as indicated by the black line transitioning from 1 to 0). By predicting the missing bounding box, the corresponding keypoints can be still extracted from the obstacle. Then the keypoints of two consecutive frames are used to generate the 3D local map points through triangulation. These local map points are subsequently projected into 2D coordinates to form matching pairs with the current keypoints. Due to

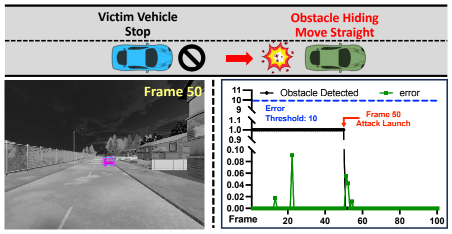


Figure 10: Obstacle hiding attack.

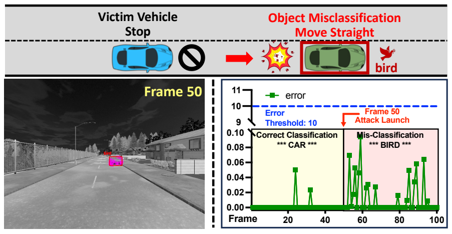


Figure 11: Target semantic confusion.

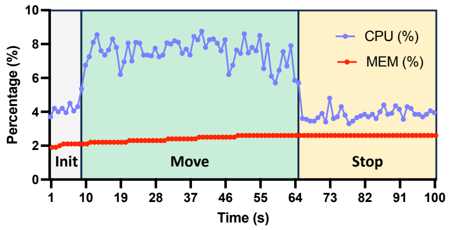


Figure 12: The performance overhead.

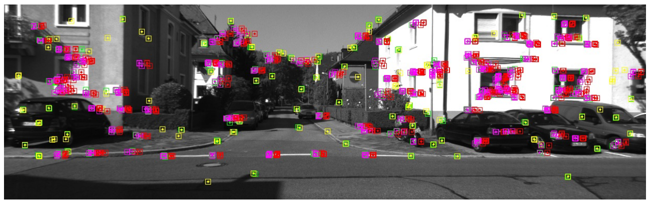


Figure 13: An example of altering keypoints attacks.

spatial continuity, the error in these matched pairs should be negligible. As the figure indicates, this error value remains significantly below the pre-set threshold of 10. Thus, the attack can be readily identified in the absence of obstacles.

Target Semantic Confusion. To implement the object misclassification attack, we manipulating the output of object classification by YOLOv4 to mislead the output of object classification. Figure 11 provides an illustration where an obstacle vehicle is positioned ahead of the victim vehicle. Under normal circumstances, the victim vehicle should decelerate and stop while recognizing the forward vehicle. However, after maliciously altering the object's class from a car to a bird, the victim vehicle would continue to drive forward since it does not recognize a flying bird as an obstacle, finally leading to a collision. Since Apollo 7 lacks an integrated YOLO system incapable of identifying types of non-obstacle objects, we have externally incorporated a third-party YOLOv4 [3] for enhanced detection capabilities. This scenario is very similar to a past accident involving a Tesla Model S, which falsely identified a white truck as the sky [1]. Figure 11 further details this event at the 50th frame where the vehicle is erroneously classified as a bird, as denoted by a red bounding box. The purple and green squares inside the boxes denote the reprojected 2D extracted keypoints and the 2D original keypoints matched against these projected keypoints, respectively.

We collected error data from total 100 successive frames preceding and following the attack, similar with the experiment conducted in the obstacle hiding attack. As depicted in Figure 11, the errors (marked by green squares) between the projected 3D map points and corresponding extracted 2D feature points of the victim vehicle are so small, considerably below our established detection threshold of 10. This can be attributed to the fact that despite the forward vehicle being misclassified as a bird, its inherent spatio-temporal consistency remains unaffected. Therefore, this characteristic can be leveraged to detect such attacks.

7.3 Adaptive Attacks Analysis

Altering Keypoints at the Frame Level. One possible adaptive attack is to directly alter or obscure the positions of extracted keypoints in a current frame, leading to anomalies in PhyScout's error values. Such a strategy must consider real-world constraints like the number and size of the patches. Therefore, the attacker must consider the density of keypoints in the current frame. High density means that keypoints are concentrated, and most keypoints can be manipulated with a small patch. However, as Figure 7-11 show, the keypoints are low in density and scattered everywhere. Thus, *we can conclude that the attacker needs to use a large number of patches simultaneously to modify the information in the image, which clearly goes against the principle of stealthiness in sensor spoofing attacks.*

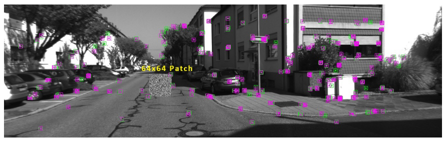
To validate our findings, we conducted a controlled experiment by manipulating the positions of extracted keypoints in a selected frame from the KITTI dataset, depicting a scene at a traffic light. In such a scenario, launching an adaptive attack is relatively easier due to the static nature of both the keypoints and map points, as the vehicle remains stationary. To maintain data consistency across the experiment, we initially preserved all data involved in the optimization process for 10 consecutive frames, which included extracted feature points, map points, and relevant parameters. Our objective was to determine the number of keypoint alterations necessary to induce false positives. Thus, we incrementally increased the number of adjusted keypoints, each shifted by 15 pixels, ensuring their eligibility as matching pairs. Preference was given to highly dense keypoints, specifically those with the most neighboring points within a 15-pixel radius around each target keypoint.

Figure 13 illustrates the minimum number of alterations needed to successfully execute such an attack. Normal map points and matching keypoints are represented by green and yellow, while the targeted victim map points and keypoints that require displacement are marked in purple and red. Our results indicate that a significant alteration exceeding 62% of the keypoints is required to impact the computed error, leading to pose inaccuracies.

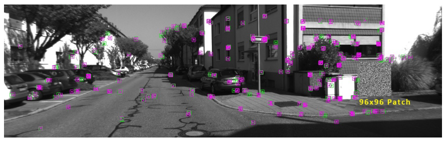
Attacking the Non-Convex Optimization Process. Another possible adaptive attack is to deceive PhyScout by manipulating perturbations using the internal gradient flow of the system. Since PhyScout relies on a non-convex function to determine mismatching errors, attackers could strategically choose positions of optimized patches within a frame to disrupt the spatio-temporal consistency by increasing the non-convex function's value.

To assess the feasibility of such an attack, we employed an optimization-based method to craft patches. For each adversarial patch, we randomly selected a position before the optimization process and repeated the whole procedure to obtain the impacts of the various patch positions in our following experiments. We chose the Adam optimizer [30] to optimize our patch, with a learning rate of 0.01 and 2,000 optimization steps. The objective during patch optimization is to increase the mean total error of each matching pair. To address non-differentiable operations, we referred to

(a) An attack case with 32x32 patch



(b) An attack case with 64x64 patch



(c) An attack case with 96x96 patch

Figure 14: An example of adversarial attack.

Table 4: The results of adversarial attacks.

Patch Size (pixels)	Mean Error	Max Error	Min Error	Variance
32x32	3.47	4.03	2.71	0.20
64x64	3.40	4.05	2.71	0.22
96x96	4.69	4.51	2.77	0.43

Table 5: Latency (ms) of each module and end-to-end system.

Module	GST	GSS	OA	OH	OM
Feature Extraction	70.1 (93%)	66.8 (92%)	50.1 (90%)	53.9 (93%)	52.4 (93%)
Match Pair Searching	0.6 (0.8%)	0.4 (0.6%)	0.5 (0.9%)	0.6 (1.0%)	0.6 (1.1%)
Offset Computation	1.1 (1.5%)	1.3 (1.8%)	0.8 (1.4%)	0.9 (1.5%)	0.8 (1.4%)
Map Points Generation	78.2	89.1	60.2	65.3	66.1
End-to-End	75.3	72.1	55.4	58.2	56.5

Table 6: Decision latency (ms) of Apollo ADAS and PhyScout pipeline based on incorrect perception states.

Scenario	PhyScout	ADAS Module			Total
	Detection	Prediction	Planning	Control	
Apollo	-	104	94	10	208
PhyScout	75	-	-	10	85

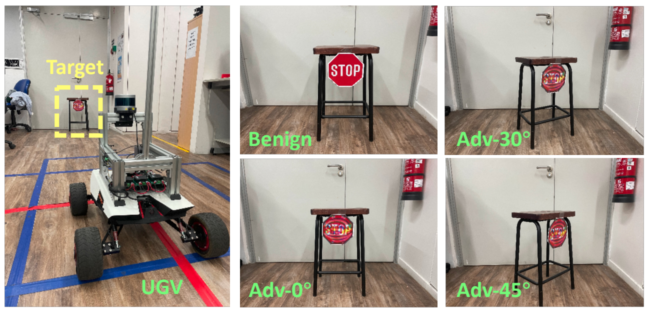


Figure 15: The setup of real-world evaluation.

BPDA [5] to transform them into mask-generation processes, i.e., we made these operations give masks to obtain the original results. This allows us to obtain the same results, which are based on non-differentiable operations, by directly applying masks to inputs. Thus, we can approximate the gradients for the patches effectively.

Figure 14 displays three types of adversarial patches generated from the 100th frame in the KITTI dataset. For each patch size, we conducted 10 runs at random positions to calculate the mean, maximum, minimum, and variance of the error, with the results presented in Table 4. We observed that almost all adversarial attacks appeared to have negligible impact on the generated error. This outcome can likely be attributed to two factors. Firstly, the uniform distribution of map points across the frame makes it challenging for a small-area patch to influence multiple matching pairs significantly. Secondly, even within the same dataset, the distribution of map points generated by each run of PhyScout may vary slightly. This variation means that such an adversarial attack has poor transferability across different map points distribution.

Table 7: Real-world evaluation results.

Scenario	Attack	Success Rate	Latency (ms)	False Positive Rate
Benign/0°	NO	N/A	N/A	0%
Benign/30°	NO	N/A	N/A	0%
Benign/45°	NO	N/A	N/A	0%
Adversarial/0°	Yes	100%	76	N/A
Adversarial/30°	Yes	100%	81	N/A
Adversarial/45°	Yes	100%	80	N/A

7.4 Performance Overhead

We evaluate PhyScout's performance overhead from two perspectives: (1) detection latency; (2) CPU and memory utilization.

Detection Latency Analysis. Table 5 presents the latency of the four main modules and the overall end-to-end latency in PhyScout. The percentages in this table indicate the latency of each key module relative to the end-to-end detection. Note that as mentioned in Section 6, the detector and mapper operate in separate working threads in parallel, and the map points generation module is off the real-time detection pipeline. The feature extraction and map points generation modules take longer time than the other two modules.

Based on the results, we obtain two conclusions. First, as a passive detection approach, PhyScout operates independently of the ADAS system. In normal scenarios without attacks, it does not affect the perception-planning-control pipeline of the baseline system, or introduce any extra latency. Second, in adversarial scenarios involving sensor spoofing attacks, PhyScout is able to promptly detect the attacks with a latency of <100ms (End-to-End Latency in Table 5). Such short detection latency allows the ADAS system to timely intervene the decision-making process, and prevent the control module from executing a malicious action caused by the attacks. This is evidenced by Table 6, with the example of Apollo: after the attack compromises the sensor data, it takes a total of 208ms to propagate to the final control module for making incorrect actions. Meanwhile, PhyScout only requires 75ms to detect the sensor data anomaly and then inform the control module to ignore the incoming malicious action (Figure 5). This shows PhyScout can effectively mitigate the sensor spoofing risks in real time.

CPU and Memory Utilization Analysis. Figure 12 depicts data collected at 1s intervals on the CPU and memory utilization over a 100s timeframe. The experiment consists of three phases: initialization of Apollo and PhyScout (0-9s), forward movement (10-65s), and stops at a traffic intersection (66-100s). We can observe that the CPU overhead is small during both the init and stop phases. The low overhead during init can be attributed to the absence of attack detection processes, while the latter is due to the reduction of new keyframes, thereby eliminating the resource demand for map point generation. The CPU overhead in the forward movement phase also remains significantly low (~8%). Meanwhile, memory utilization remained constant at a 2%. This overhead can be almost negligible compared with other defense frameworks with deep learning models.

7.5 Real-world Case Study

Setup Up. A typical case study of obstacle hiding experiment with an Unmanned Ground Vehicle (UGV) was conducted (Figure 15). The UGV is equipped with an Intel RealSense D435i front-facing camera, capturing 1080p images at 30fps, as well as a Bosch BMI055 6-axis IMU integrated on an onboard chip. Our experimental setup employs both benign and adversarial stop signs to implement an obstacle hiding attack [38], which are color-printed and strategically positioned on a scaled-down, straight roadway delineated by colored tape. The UGV is consistently positioned at an initial distance of 5m from the target sign and instructed to proceed in a straight path. To ensure safety within the confines of the indoor setting, the initial velocity is regulated at 1m/s at the commencement of each trial. We replicate this experimental procedure for twenty trials, each comprising three benign and three adversarial signs, angled diversely for each run.

Result. Table 7 presents our findings obtained in a real-world setting. Notably, the benign stop sign does not initiate the Attack signal in any observed instances. In contrast, all adversarial patches, irrespective of their angular orientations, achieve a 100% detection rate with a latency within 1s across all 10 trials. Further, PhyScout demonstrates a zero false positive rate in benign scenarios, attributable to the continual maintenance of spatio-temporal consistency. This experiment substantiates the occurrence of previously hypothesized spatio-temporal consistency conflicts.

8 CONCLUSION

This paper presents PhyScout, an innovative defense framework that mitigates the impacts of sensor spoofing attacks on AVs. By inspiring from the human observers, PhyScout introduces a novel approach to detecting these attacks via spatio-temporal consistency. Through the generalization of conflicts within sensor spoofing attacks and the formalization of these conflicts into a least squares problem. Our approach, utilizing image-based feature point extraction and matching to design risk identification methods, has been thoroughly tested in various environments. In comparison with existing frameworks, PhyScout significantly outperforms in terms of detection delay, performance overhead, and conflict visualization.

ACKNOWLEDGMENTS

This research/project is supported by the NTU-Desay SV Research Program under Grant 2018-0980, the National Research Foundation, Singapore and DSO National Laboratories under its AI Singapore Programme (AISG Award No: AISG2-GC-2023-008).

REFERENCES

[1] 2016. How Tesla's autopilot got it wrong in fatal crash. https://www.afr.com/technology/how-teslas-autopilot-got-it-wrong-in-fatal-crash-20160704-gpxsje.

[2] 2020. An Open Source Self-Driving Car. https://www.udacity.com/self-driving-car/.

[3] 2023. Yolo 4. https://github.com/AlexeyAB/darknet.

[4] Pablo Fernández Alcantarilla, Jesús Nuevo, and Adrien Bartoli. 2013. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In *British Machine Vision Conference.*

[5] Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In *ECCV.*

[7] Berta Bescós, José M. Fácil, Javier Civera, and José Neira. 2018. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robotics Autom. Lett* (2018).

[8] Rohit Bhatia, Vireshwar Kumar, Khaled Serag, Z. Berkay Celik, Mathias Payer, and Dongyan Xu. 2021. Evading Voltage-Based Intrusion Detection on Automotive CAN. In *NDSS.*

[9] Gedare Bloom. 2021. WeepingCAN: A Stealthy CAN Bus-off Attack. In *NDSS Workshop on Automotive and Autonomous Vehicle Security.*

[10] Alessandro Calò, Paolo Arcaini, Shaukat Ali, Florian Hauer, and Fuyuki Ishikawa. 2020. Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems. In *ICST.*

[11] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *IEEE Transactions on Robotics* (2021).

[12] 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 S&P.*

[13] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. [n. d.]. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. In *CCS 2019.*

[14] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng (Polo) Chau. 2018. ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector. In *ECML/PKDD.*

[15] Zhiyuan Cheng, James Liang, Hongjun Choi, Guanhong Tao, Zhiwen Cao, Dongfang Liu, and Xiangyu Zhang. 2022. Physical attack on monocular depth estimation with optimal adversarial patche. In *ECCV.*

[16] Gelei Deng, Yuan Zhou, Yuan Xu, Tianwei Zhang, and Yang Liu. 2021. An Investigation of Byzantine Threats in Multi-Robot Systems. In *International Symposium on Recent Advances in Intrusion Detection.*

[17] Vishal Dey, Vikramkumar Pudi, Anupam Chattopadhyay, and Yuval Elovici. 2018. Security Vulnerabilities of Unmanned Aerial Vehicles and Countermeasures: An Experimental Study. In *International Conference on VLSI Design.*

[18] Ranjie Duan, Xiaofeng Mao, A. Kai Qin, Yuefeng Chen, Shaokai Ye, Yuan He, and Yun Yang. 2021. Adversarial Laser Beam: Effective Physical-World Attack to DNNs in a Blink. In *CVPR.*

[19] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Amir Rahmati Bo Li, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *CVPR.*

[20] Xiang Gao, Tao Zhang, Yi Liu, and Qinrui Yan. 2017. *14 Lectures on Visual SLAM: From Theory to Practice.* Publishing House of Electronics Industry.

[21] R. Spencer Hallyburton, Yupei Liu, Yulong Cao, Z. Morley Mao, and Miroslav Pajic. 2021. Security Analysis of Camera-LiDAR Fusion Against Black-Box Attacks on Autonomous Vehicles. In *CoRR.*

[22] Xingshuo Han, Guowen Xu, Yuan Zhou, Xuehuan Yang, Jiwei Li, and Tianwei Zhang. 2022. Physical backdoor attacks to lane detection systems in autonomous driving. In *ACMMM.*

[23] Daojing He, Hong Liu, Sammy Chan, and Mohsen Guizani. 2019. How to Govern the Non-Cooperative Amateur Drones? *IEEE Network* (2019).

[24] Daojing He, Yinrong Qiao, Shiqing Chen, Xiao Du, Wenjie Chen, Sencun Zhu, and Mohsen Guizan. 2019. A Friendly and Low-Cost Technique for Capturing Non-Cooperative Civilian Unmanned Aerial Vehicles. *IEEE Network* (2019).

[25] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L. Yuille, Changqing Zou, and Ning Liu. 2020. Universal Physical Camouflage Attacks on Object Detectors. In *CVPR.*

[26] Yunhan Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei. 2020. Fooling Detection Alone is Not Enough: Adversarial Attack against Multiple Object Tracking. In *International Conference on Learning Representations.*

[27] Zizhi Jin, Xiaoyu Ji, Yushi Cheng, Bo Yang, Chen Yan, and Wenyuan Xu. 2023. PLA-LiDAR: Physical Laser Attacks against LiDAR-based 3D Object Detection in Autonomous Vehicle. In *IEEE Symposium on Security and Privacy.*

[28] 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 Symposium.*

[29] Hyungsub Kim, Muslum Ozgur Ozmen, Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu. 2021. PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles. In *NDSS.*

[30] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR Poster.*

[31] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. 2020. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. In *CVPR.*

[32] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. 2021. CANNON: Reliable and Stealthy Remote Shutdown Attacks via Unaltered Automotive Microcontrollers. In *IEEE S&P.*

[33] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. G2o: A general framework for graph optimization. In *IEEE ICRA.*

[34] Alexander Levine and Soheil Feizi. [n. d.]. (De)Randomized Smoothing for Certifiable Defense against Patch Attacks. In *NeurIPS 2020*.

[35] Wan-Yi Lin, Fatemeh Sheikholeslami, Jinghao Shi, Leslie Rice, and J Zico Kolter. 2021. Certified robustness against physically-realizable patch attack via randomized cropping. *ICLR Open Review* (2021).

[36] Shinan Liu, Xiang Cheng, Hanchao Yang, Yuanchao Shu, Xiaoran Weng, Kexiong Curtis Zeng Ping Guo, Gang Wang, and Yaling Yang. 2020. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing. In *USENIX Security Symposium*.

[37] 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 Symposium*.

[38] 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 Symposium*.

[39] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* (2004).

[40] Mulong Luo, Andrew C. Myers, and G. Edward Suh. 2020. Stealthy Tracking of Autonomous Vehicles with Cache Side Channels. In *USENIX Security Symposium*.

[41] Yanmao Man and Ming Li. 2020. GhostImage: Remote Perception Attacks against Camera-based Image Classification Systems. In *International Symposium on Recent Advances in Intrusion Detection*.

[42] Yanmao Man, Raymond Muller, Ming Li, Z. Berkay Celik, and Ryan Gerdes. 2023. That Person Moves Like A Car: Misclassification Attack Detection for Autonomous Systems Using Spatiotemporal Consistency. In *USENIX Security Symposium*.

[43] Jan Hendrik Metzen and Maksym Yatsura. 2021. Efficient Certified Defenses Against Patch Attacks on Image Classifiers. In *ICLR*.

[44] Noriyuki Miura, Tatsuya Machida, Kohei Matsuda, Makoto Nagata, Shoei Nashimoto, and Daisuke Suzuki. 2019. A Low-Cost Replica-Based Distance-Spoofing Attack on mmWave FMCW Rada. In *ASHES@CCS*.

[45] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robotics* (2015).

[46] Raul Mur-Artal and Juan D. Tardós. 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics* (2017).

[47] Shoei Nashimoto, Daisuke Suzuki, Takeshi Sugawara, and Kazuo Sakiyama. 2018. Sensor CON-Fusion: Defeating Kalman Filter in Signal Injection Attack. In *ACM AsiaCCS*.

[48] 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*.

[49] Ben Nassi, Dudi Nassi, Raz Ben-Netanel, Yisroel Mirsky, Oleg Drokin, and Yuval Elovici. 2020. Phantom of the ADAS: Phantom Attacks on Driver-Assistance Systems. *IACR Cryptology ePrint Archive* (2020).

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

[51] Davide Quarta, Marcello Pogliani, Mario Polino, Federico Maggi, Andrea Maria Zanchettin, and Stefano Zanero. 2017. An Experimental Security Analysis of an Industrial Robot Controller. In *IEEE S&P*.

[52] Raul Quinonez, Jairo Giraldo, Luis E. Salazar, Erick Bauman, Alvaro A. Cárdenas, and Zhiqiang Lin. 2020. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *USENIX Security Symposium*.

[53] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *IEEE ICCV*.

[54] Harshad Sathaye, Martin Strohmeier, Vincent Lenders, and Aanjhan Ranganathan. 2022. An Experimental Study of GPS Spoofing and Takeover Attacks on UAVs. In *USENIX Security Symposium*.

[55] 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 Symposium*.

[56] Desmond Allan Schmidt, Kenneth Radke, Seyit Ahmet Çamtepe, Ernest Foo, and Michal Ren. 2016. A Survey and Analysis of the GNSS Spoofing Threat and Countermeasures. *Comput. Surveys* (2016).

[57] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Reinforcement Learning for Autonomous Driving. In *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*.

[58] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. 2017. Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications. In *Cryptographic Hardware and Embedded Systems*.

[59] Siciliano, Bruno, and Oussama Khatib. 2016. *Springer handbook of robotics*. Springer, Secaucus, NJ, USA: Sprinter-Verlag New York, Inc.

[60] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. 2018. Physical Adversarial Examples for Object Detectors. In *Workshop on Offensive Technologies*.

[61] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. 2020. Towards Robust LiDAR-based Perception in Autonomous Driving: General Black-box Adversarial Sensor Attack and Countermeasures. In *USENIX Security Symposium*.

[62] Zhi Sun, Sarankumar Balakrishnan, Lu Su, Arupjyoti Bhuyan, Pu Wang, and Chunming Qia. 2021. Who Is in Control? Practical Physical Layer Attack and Defense for mmWave-Based Sensing in Autonomous Vehicles. *IEEE Transactions on Information Forensics and Security* (2021).

[63] Sebastian Thrun, Wolfram Burgard, and Diter Fox. 2005. *Probabilistic Robotics*. The MIT Press.

[64] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In *ICSE*.

[65] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *EuroSP*.

[66] Yazhou Tu, Zhiqiang Lin, Insup Lee, and Xiali Hei. 2018. Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors. In *USENIX Security Symposium*.

[67] Ziwen Wan, Junjie Shen, Jalen Chuang, Xin Xia, Joshua Garcia, Jiaqi Ma, and Qi Alfred Chen. 2022. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *NDSS*.

[68] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. 2021. Dual Attention Suppression Attack: Generate Adversarial Camouflage in Physical World. In *CVPR*.

[69] 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*.

[70] Zhengbo Wang, Kang Wang, Bo yang, Shangyuan Li, and Aimin Pan. 2017. Sonic gun to smart devices. In *Black Hat USA*.

[71] Zuxuan Wu, Ser-Nam Lim, Larry S. Davis, and Tom Goldstein. 2020. Making an Invisibility Cloak: Real World Adversarial Attacks on Object Detectors. In *European Conference on Computer Vision*.

[72] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. 2021. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. In *USENIX Security Symposium*.

[73] Chong Xiang and Prateek Mittal. 2021. DetectorGuard: Provably Securing Object Detectors against Localized Patch Hiding Attacks. In *ACM CCS*.

[74] Chong Xiang and Prateek Mittal. 2021. PatchGuard++: Efficient Provable Attack Detection against Adversarial Patches. In *CoRR abs/2104.12609*.

[75] Chaowei Xiao, Ruizhi Deng, Bo Li, Taesung Lee, Benjamin Edwards, Jinfeng Yi, Dawn Song, Mingyan Liu, and Ian M. Molloy. 2019. AdvIT: Adversarial Frames Identifier Based on Temporal Consistency in Videos. In *IEEE ICCV*.

[76] 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*.

[77] Yuan Xu, Xingshuo Han, Gelei Deng, Guanlin Li, Yang Liu, Jiwei Li, and Tianwei Zhang. 2023. SoK: Rethinking Sensor Spoofing Attacks against Robotic Vehicles from a Systematic View. In *EuroSP*.

[78] Yuan Xu, Tianwei Zhang, and Yungang Bao. 2021. Analysis and Mitigation of Function Interaction Risks in Robot Apps. In *International Symposium on Recent Advances in Intrusion Detection*.

[79] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. [n. d.]. SoK: A Minimalist Approach to Formalizing Analog Sensor Security. In *IEEE S&P 2021*.

[80] Chen Yan, Wenyuan Xu, and Jianhao Liu. 2016. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. In *Def Con*.

[81] Chen Yan, Zhijian Xu, Zhanyuan Yin, Xiaoyu Ji, and Wenyuan Xu. 2022. Rolling Colors: Adversarial Laser Exploits against Traffic Light Recognition. *CoRR* (2022).

[82] Ping yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. 2020. Certified Defenses for Adversarial Patches. In *ICLR*.

[83] Chengzeng You, Zhongyuan Hau, and Soteris Demetriou. 2021. Temporal Consistency Checks to Detect LiDAR Spoofing Attacks on Autonomous Vehicle Perception. In *Workshop on Security and Privacy for Mobile AI*.

[84] Kexiong (Curtis) Zeng, Shinan Liu, Yuanchao Shu, Dong Wang, Haoyu Li, Yanzhi Dou, Gang Wang, and Yaling Yang. 2018. All Your GPS Are Belong To Us: Towards Stealthy Manipulation of Road Navigation Systems. In *USENIX Security Symposium*.

[85] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *ASE*.

[86] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. 2019. Seeing isn't Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors. In *CCS*.

[87] Ce Zhou, Qiben Yan, Yan Shi, and Lichao Sun. 2022. DoubleStar: Long-Range Attack Towards Depth Estimation based Obstacle Avoidance in Autonomous Systems. In *USENIX Security Symposium*.

[88] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2021. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving?. In *ACM CCS*.