Exorcising "Wraith": Protecting LiDAR-based Object Detector in Automated Driving System from Appearing Attacks

Qifan Xiao*, Xudong Pan*, Yifan Lu, Mi Zhang✉, Jiarun Dai, Min Yang✉
Fudan University, China
{20210240056@, xdpan18@, luyifan21@m., jrdai14@, mi_zhang@, m_yang@}fudan.edu.cn
(: co-first authors; ✉: corresponding authors)*

Abstract

Automated driving systems rely on 3D object detectors to recognize possible obstacles from LiDAR point clouds. However, recent works show the adversary can forge non-existent cars in the prediction results with a few fake points (i.e., *appearing attack*). By removing statistical outliers, existing defenses are however designed for specific attacks or biased by predefined heuristic rules. Towards more comprehensive mitigation, we first systematically inspect the mechanism of recent appearing attacks: Their common weaknesses are observed in crafting fake obstacles which (i) have obvious differences in the local parts compared with real obstacles and (ii) violate the physical relation between depth and point density.

In this paper, we propose a novel plug-and-play defensive module which works by side of a trained LiDAR-based object detector to eliminate forged obstacles where a major proportion of local parts have low *objectness*, i.e., to what degree it belongs to a real object. At the core of our module is a *local objectness predictor*, which explicitly incorporates the depth information to model the relation between depth and point density, and predicts each local part of an obstacle with an *objectness* score. Extensive experiments show, our proposed defense eliminates at least 70% cars forged by three known appearing attacks in most cases, while, for the best previous defense, less than 30% forged cars are eliminated. Meanwhile, under the same circumstance, our defense incurs less overhead for AP/precision on cars compared with existing defenses. Furthermore, We validate the effectiveness of our proposed defense on simulation-based closed-loop control driving tests in the open-source system of Baidu's Apollo.

1 Introduction

In automated driving systems (ADS), multiple deep neural networks (DNNs) are jointly deployed to provide key functionalities of localization, perception and planning, stimulating the recent development of automated transportation [8, 33, 36]. The robustness of each DNN module is of key importance to the security of the whole ADS. A typical example is the *perception* module, which relies on a vector of *object detectors*, based on multiple sources like cameras and LiDARs [7], to predict the categories and locations of the obstacles around the ADS [12, 32]. As the LiDAR point clouds (PCs) contain richer location information than the images from cameras, most commercial ADS, including Google's Waymo One [5,6] and Baidu's Apollo [1,2], set LiDARs as the main sensors and rely on the detection results of *LiDAR-based object detectors* for obstacle perception [27, 32, 37, 38].

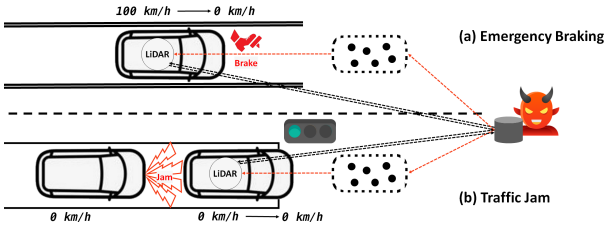

Figure 1: Appearing attacks on LiDAR-based object detectors in ADS can cause severe traffic accidents by forging cars.

Differently using PC as the model input, LiDAR-based object detectors still share the common vulnerability against *adversarial examples* [12, 42, 48]. In general, the attacker can spoof the LiDAR sensors with a limited number of perturbed/crafted points to mislead the detector's prediction. As a popular attack class, the *appearing attack* aims at forging non-existent cars in the detection results to cause traffic jams and emergency braking [11, 39] (Fig.1). Despite the severity, existing defenses [15, 24, 41] either have strong prior assumptions on the undergoing attacks, or are biased by predefined heuristic rules, insufficient for handling complex driving scenarios (§3.3).

Our Work. In this paper, we propose a novel plug-and-play defense for 3D object detectors, which, instead of constructing a more robust model, adopts a *local objectness predictor* (LOP) module to detect and eliminate forged obstacles from the original detection results. In general, our LOP is designed

as a point-wise PC classifier [29, 34, 35, 45] which learns to predict each local part of a detected object with an *objectness* score, i.e., the confidence of whether the local part belongs to a real object. By systematizing recent appearing attacks, we develop the following defensive insights:

1. Recent appearing attacks focus on increasing the confidence score of a fake detection result without considering the local difference between a real and a forged obstacle. Although an increased confidence score enhances the possibility of a non-existent obstacle to be detected by a 3D object detector, most appearing attacks leave the fake and the real obstacles locally distinguishable when inspected at the granularity of pillars or voxels (§4.1)

2. Constrained by the physical capability of attack apparatus, appearing attacks are usually unable to forge a fake obstacle without violating some physical laws, especially the inimitable relation between the depth and the point density of real obstacles [14]. To pose real-world threats, the forged obstacles have to be close to the victim ADS, because otherwise they can be easily bypassed after the victim's re-routing. Yet, constrained by the attack apparatus (e.g., a laser transmitter [41]), the attacker can only forge a limited number of points near the victim during one scan of the LiDAR, which could hardly reach the normal point density of a real car at a close distance (§4.2).

Concurrent to our defense, Hau et al. [24] also notices the importance of the physical law in detecting forged obstacles, and presents a set of hand-crafted rules to eliminate the anomaly. Our work steps further by showing stronger robustness can be achieved if we exploit learning-based techniques to model the complicated physical laws. In fact, modeling the relation between the depth and the point density is rather challenging with hand-crafted rules. For example, although most of the real cars with smaller depth tend to have larger point density, those real cars occluded by others may also have smaller depth and point density simultaneously (Fig.4). To address this challenge, we implement the LOP as a DNN-based point-wise PC classification model and explicitly incorporate the depth information of each point into its feature vector. This substantially improves the modeling capability compared with using the original input feature for statistical outlier detection.

Moreover, another technical challenge is the lack of no explicit annotation available for supervising the training of LOP in standard 3D object detection datasets. Inspired by a recent observation that a single part of the input already contains rich semantic information for a PC model to predict its related object's category and location [15], we construct a self-supervised learning task where the LOP learns to predict whether a pillar intersects with any bounding box of real objects based on the features of its inside points. During the detection, we first divide the input 3D space into equal-sized pillars, then the LOP predicts an objectness score for each pillar intersected with a predicted object's bounding box. By majority voting on the local objectness predictions, our defense determines whether the object is real or fake (§4.3).

Our Contributions. In summary, the key contributions of this work are as follows:

- We systematize the limitations of recent appearing attacks in violating the physical invariants and propose a learning-based defense to detect the forged obstacles with anomaly in the relation between the depth and the point density for the mainstream LiDAR-based object detectors.
- We propose the design of our local objectness predictor (LOP) which learns to predict the confidence of whether a local object part belongs to a real object, and allows plug-and-play integration with different defense targets for enhancing robustness against popular appearing attacks.
- Extensive evaluation on mainstream 3D detectors (i.e., PointPillars [27], PointRCNN [38] and PV-RCNN [37]) on the KITTI dataset [19] and on real-world PC data we collect from a driving test of the D-KIT Advanced with Velodyne-128 [4] validate the advantages of our proposed defense under three popular attacks. For example, with the same-level trade-off in model utility, our proposed defense eliminates at least 70% cars forged by most appearing attacks, while the best baseline method only eliminates the forged ones less than 30%.
- Moreover, we empirically validate that the effectiveness of our proposed LOP is robust to the architecture design of the LOP, the type of the defense target (including fusion models) which further implies our defense is more general-purpose than existing defenses. Besides, we also provide a preliminary study on the robustness of LOP against adaptive attacks.
- We further implement and evaluate the effectiveness of LOP in Apollo 6.0.0, an end-to-end open-source self-driving system, with closed-loop control in the LGSVL simulation tests, which validates the system-level usefulness of our proposed defense in both benign and adversarial scenarios.

2 Background

Basics of LiDAR. As one of the main sensors deployed in an automated driving system (ADS), a LiDAR (Light Detection and Ranging) scans the surrounding environment and generates a point cloud (PC) $X = \{(x_i, y_i, z_i, int_i)\} \in R^{n\times 4}$, including n points with (x_i, y_i, z_i) as i-th point's location and int_i as i-th point's intensity, during each detection [7, 39]. Technically, the LiDAR first emits a laser ray consecutive in both horizontal and vertical directions, which then captures the reflected lasers, records their time of flight and light intensity, and further computes the depth and 3D coordinate of the points related to these reflected lasers. Finally, the LiDAR collects these information to generate the raw PC, which represents the object surfaces in the surrounding environment, and sends this raw PC to the ADS for downstream processing.

3D Object Detectors. DNN-based 3D object detectors empower modern ADS for perceiving and detecting objects in

the surrounding environment (i.e., *obstacle perception*). Technically, a 3D object detector usually takes PC as the input and returns the category and *bounding box*, a rectangle or cuboid which bounds the detected object to represent its location in a PC, of each perceived object [21]. In most cases, 3D object detectors can be regarded as the combination of three modules: the preprocessing, the backbone and the prediction modules.

A typical preprocessing module first divides the points of PC into a number of sets (e.g., voxels or pillars) based on specific rules and then calculates the statistical information [27, 52], or uses DNN models, such as PointNet [34] or DGCNN [45], to generate the feature vectors for each point [38]. Then, the backbone module implemented with 2D/3D convolutional neural networks (CNN) [26,28] extracts the PC's features and generates the global feature map. Finally, the prediction module in one-stage 3D object detectors like VoxelNet [52] and PointPillars [27] directly predicts the bounding box and category of each obstacle based on the global feature map. Differently, in two-stage 3D object detectors like PointRCNN [38] and PV-RCNN [37], the prediction module predicts the proposal bounding boxes of objects based on the global feature map and generate a local feature map for each object based on the combination of the global feature map and the related proposal bounding boxes at the first stage, and then the final bounding box and category of each obstacle based on each local feature map at the second stage.

Adversarial Example. In general, given a machine learning model F and a normal sample x with label y, an adversarial example x' is generated from x by adding a slight perturbation to mislead the victim model's prediction while causing no modification to either the model's architecture or the parameters [23,42,49]. According to the attack goal, an adversarial example can be further categorized into untargeted and targeted. By definition, an untargeted attack aims at misleading the victim model into $F(x') \neq y$, while a targeted attack aims at misleading the victim model into $F(x') = y'$, where y' is the target label specified by attacker. According to [13], the targeted adversarial attack can be further represented as the optimization problem:

$$argmin_{x'} ||x - x'||_p \quad \text{s.t.} \quad F(x') = y' \text{ and } x' \in X \qquad (1)$$

where the objective $\min \|x - x'\|$ restricts the region of perturbation (i.e., attack budget) and X denotes the input space. In the context of ADS, to cause severe safety issues, several recent adversarial attacks focus on conducting LiDAR spoofing to forge a non-existent object in the detection results of a LiDAR-based object detector, or called *appearing attacks*, on which we provide a detailed review in Section 3.2.

3 Security Settings

3.1 Threat Model

• **Attacker's Goal**. In general, the direct goal of an appearing attack is to forge fake cars, in the detection results of the LiDAR-based object detector in ADS. To refine the attack goal above, we first analyze the following two attack scenarios of an appearing attack.

Attack Scenario 1. *(On the Highways)* As shown in the top part (a) of Fig.1, an attacker can spoof the LiDAR of the victim ADS when it passes by. Detecting a forged car at the immediate front, the victim will make a stop decision and decrease its speed to 0 km/h within seconds. The unpredictable emergency braking may leave no reaction time for other vehicles behind. This may lead to a rear-end collision or even more severe traffic accidents.

Attack Scenario 2. *(At the Traffic Lights)* Similarly, as shown in the bottom part (b) of Fig.1, the attacker conducts LiDAR spoofing when the victim ADS stops at the red light. By forging a fake car ahead, the victim will keep immobile even after the traffic signal turns green, blocking other vehicles behind and causing a traffic jam.

As the two attack scenarios show, to cause a real-world threat, the forged cars are required to be not only recognized by LiDAR-based object detectors with sufficiently large confidence scores, but also close enough to result in the re-routing of the victim. Therefore, we further refine the attack goal to expect the cars to be forged in a close distance to the victim. Specifically, in this work we require a forged car to be within a $5 \sim 10$ meters to the victim to pose a sufficient threat [11,41].

• **Attacker's Capability.** Following the threat model in recent attacks [11, 41], our defense mainly aims at mitigating an attacker satisfying the following threat model:

Assumption 1. *(Prior Knowledge)* The attacker knows the architecture and the parameters of the LiDAR-based object detector deployed on the victim ADS (i.e., *white-box*).

Assumption 2. *(Number of Added Points)* The attacker can inject at most 200 points (according to [41]) into the input PC of the victim 3D object detector in one scan of LiDAR.

Assumption 3. *(Features of Added Points)* The attacker is allowed to inject points at any location and with arbitrary light intensity, which is imposed for a more generic defense.

• **Attack Process.** Before the attack starts, the attacker deploys a physical equipment to receive the lasers emitted by the victim ADS's LiDAR, and shoot lasers back to the LiDAR. Later, the LiDAR-based 3D object detectors of the victim takes the infected PC and predicts a non-existent car. Finally, the victim re-routes to avoid the non-existent car, which may lead to severe collision accidents.

3.2 Recent Appearing Attacks

Next, we review the recent appearing attacks on LiDAR-based object detectors. As one of the earliest work, Shin et al. propose a spoofing attack by randomly injecting points into a certain area regardless of the LiDAR-based object detectors of the victim ADS, which is sufficient to forge a non-existent car [39]. Inspired by Shin's work, Cao et al. standardize the attack pipeline of adversarial spoofing attack, and propose

an appearing attack, Adv-LiDAR, which aims at breaking Apollo's detection system [11]. By modeling the preprocessing and postprocessing modules in Apollo's LiDAR-based object detector, Adv-LiDAR successfully uses traditional adversarial attack technology to forge non-existent cars. However, Sun et al. later prove that other 3D object detectors such as PointPillars and PointRCNN will not be affected by the the adversarial samples generated by Adv-LiDAR, and then suggest a more general black-box appearing attack based on the intrinsic physical nature of LiDARs [41]. Also, another attack by Yang et al. shares the same attack goal as the above appearing attacks but uses a different attack process and physical equipment [48]. Specifically, they use a physical object which is specially designed to tempt the 3D object detector to predict itself as a car with a falsely enlarged bounding box and therefore fabricate a non-existent part of this object in the model's perception. For completeness, we also cover this attack into the appearing attacks in experiments.

3.3 Previous Defenses

• **Rationale behind Defenses by Elimination**. To eliminate the forged vehicles crafted by appearing attacks, a defense would unavoidably remove a small ratio of detected real objects from the prediction of 3D object detectors. However, we argue this would hardly cause as substantial damages to the ADS as the mistake of detecting forged vehicles. It is mainly because: (i) As described in the attacker's goal, obstacles which appear near the ADS take the most decisive effect on the vehicle's future planning. Therefore, incorrect elimination of a real obstacle far from this vehicle may have limited influence on the decision-making of the ADS [41]. (ii) In ADS, the *multi-object tracking* (MOT) module which follows the perception module will take the predictions from the LiDAR-based object detectors as input, maintain and predict the trajectories of objects nearby [17,31,46]. By design, MOT usually creates an object trajectory for a newly predicted object which is constantly detected for 6 frames, while removes an overdue object trajectory which is continuously unmatched with any predicted objects for 60 frames in a common visual perception system [53] of 30 FPS. This mechanism guarantees that it is much easier for an ADS to create a fake object in its perception due to a successful appearing attack than forgetting a real object, due to the occasional misprediction of the LiDAR-based detector itself or the incorrect elimination of some real objects by such a defense.

Therefore, it is reasonable to tolerate a small ratio of false alarms from defenses by elimination and recognize the importance of defending against appearing attacks by slightly trading the recall of LiDAR-based object detectors. However, the existing defense methods which are possibly against appearing attacks remain limitations in their design, so it is hard for them to maintain good performance in different scenes. To make it clear, we further analyze these defense methods and discuss their limitations accordingly.

• **Limitations of Universal Defenses.** SRS (Simple Random Sampling) and SOR (Statistical Outlier Removal) are two universal defense methods for PC models. They are both unaware of attacks and against adversarial attacks by removing suspect points in input PC.

(1) SRS. SRS is in essence a random method regardless of any auxiliary information [51]. Formally speaking, given a raw input PC X with n points, SRS will randomly sample $M(M < n)$ points from X by $P(X) = \{\mathbb{I}_x | x \in X,\ \mathbb{I}_x \sim Bernoulli(0.5)\}$, where $\mathbb{I}_x$ indicates the existence of each point x in X.

(2) SOR. For a raw input PC X, SOR computes the average of the k-nearest neighbors' (kNN) distances for each point in X, and counts the mean μ and the standard deviation σ of these distances. Then, it recognize those points which fall outside the range of $[\mu - \alpha \cdot \sigma, \mu + \alpha \cdot \sigma]$ as noises and removes them from X, where $\alpha = 1.1$ is its hyper-parameter [51].

• **Limitations of Specific Defenses.** CARLO (oCclusion-Aware hieRarchy anomaLy detectiOn), SVF (Sequential View Fusion) and Shadow-Catcher are three specific heuristic defense methods for 3D object detectors. They both specify the attack as a black-box appearing attack proposed in Sun's work [41], and perform defense by removing suspect points in input PC or deleting suspect objects in the final prediction.

(1) CARLO. CARLO is a heuristic defense algorithm proposed by Sun et al. to detect the cars forged by their black-box appearing attack [41]. For each object predicted by the 3D object detectors, CARLO computes an anomalous ratio r in one of the following two ways: (1) **FSD (Free Space Detection)**, which defines $r = \sum_{c \in S^c} FC(c)/|S^c|$, where S^c is a set including all the cells in this object's bounding box, and $FC(c)$ is a $0/1$ function indicating whether there are input points in the cell c; and (2) **LPD (Laser Penetration Detection)**, which defines $r = |S\downarrow^p|/|S\downarrow^p \bigcup S^p \bigcup S\uparrow^p|$, where the superscript p indicates the corresponding set is composed of points. Specifically, $S\downarrow^p$ contains the input points in the space behind this object, S^p is contains the input points inside this object's bounding box, and $S\uparrow^p$ contains the input points in the space between this object and the LiDAR.

Then, CARLO compares all these r with a fixed threshold R. For those objects with $r > R$, CARLO recognizes them as fake objects and erases them from the prediction.

(2) SVF. Similarly, SVF is another defense algorithm suggested by Sun et al., but its key is more similar to SOR: removing outliers from the raw input PC. As an extra end-to-end network, SVF turns the raw input PC into front-view (FV) representation and uses LU-Net [9], a PC segmenter, to calculate a segmentation score for each point. SVF then concatenates these scores with their related points' input features to regenerate the input PC, and passes this augmented PC to the 3D object detector as input.

(3) Shadow-Catcher. As our concurrent work, Shadow-Catcher [24] also exploits the physical law to improve the robustness of the 3D object detectors in self-driving system.

However, Shadow-Catcher is mainly based on hand-crafted rules to determine the forged obstacles, while our work proposes the first learning-based defense scheme to model the complicated physical relation between the depth and density of real objects for defensive purposes. Specifically, Shadow-Catcher computes an anomaly score for each detected object based on the distances of the points inside its bounding box to four key lines related to its bounding box, then compare this score with a presetting threshold to determine whether the perceived obstacle is forged.

As a final remark, most of the previous defenses are initially designed for mitigating specific appearing attacks. In this sense, the performance of previous defenses against each popular attack remains unjustified in a systematic way, which we accomplish in our evaluation.

4 Defense with Local Objectness Predictor

Methodology Overview. As shown in Fig.2, the pipeline of our proposed defense can be divided into three stages: training sample generation, objectness predictor construction and fake object elimination. In the training sample generation stage, we construct a learning task for our local objectness predictor (LOP), which consists of pairs of points inside a small local pillar and its corresponding objectness label, annotated in a fully self-supervised way without additional annotation except for a standard training dataset for LiDAR-based object detectors. Then, in the objectness predictor construction stage, we train the LOP to learn to predict the objectness score for each pillar, i.e., the confidence of whether a local part belongs to a real object. Finally, in the fake object elimination stage, we use our trained LOP to predict an objectness score for each small pillar intersected with the bounding boxes of the predicted objects, and determine whether these objects are real by majority voting. Below, we elaborate on the insights and the technical designs in each stage of our defense.

4.1 Training Sample Generation

4.1.1 Insight: Global Objectness $\neq$ Local Objectness

By inspecting the design of recent appearing attacks, we observe that most attacks focus on increasing the confidence scores of the forged obstacles, which represents the possibility of the detected object to be real. Equivalently, according to our definition of objectness, the confidence score can be explained as a *global objectness score* related with the predicted obstacle to some extent. As most LiDAR-based object detectors by design keep those objects with higher confidence, or global objectness scores, in their final predictions, increasing confidence scores is the most direct way for the attacker to successfully forge a non-existent obstacle. However, to increase the global objectness score of a forged obstacle does not necessarily lead to a higher objectness score for each local part. With the following experiments, we observe that most of the recent appearing attacks have ignored the local difference, i.e. the spatial distance of two corresponding subsets, between a real and a forged obstacle, which leaves an exploitable trace for the defender.

• **A Pilot Study.** As the description in Section 3, the mainstream appearing attacks all focus on forging cars, so we mainly validate the above observation on cars. We first randomly sample one real car from the training set of KITTI [19] and $1,000$ forged cars crafted by three mainstream appearing attacks [11, 41, 48] (later described in Section 5). Next, we translate the interior points of each ground-truth car and each forged car into its local coordinate system, rotated by the lead angle to the identical orientation. Then, for the point set S of the real car and S' of each forged car, we measure the distance between them by using the chamfer distance [47] and the average square L2 distance of kNN as metrics.

Specifically, we collect points belonging to the real car as S_R. For each forged car, we first collect points belonging to it as S_F. Then, we split the point space into equal-sized pillars p_j (as in Fig.2), and generate a point subset $S_{F,j} = S_F \cap p_j$ for each pillar. Finally, we calculate the global difference and local difference as follows:

$$D_{\text{global}} = D(S_R, S_F) \tag{2}$$

$$D_{\text{avg\_local}} = \frac{1}{|\{S_{F,j}\}|} \sum D(S_R, S_{F,j}) \tag{3}$$

$$D_{\text{half\_max\_local}} = \frac{1}{N_{\text{half}}} \text{Top}_{N_{\text{half}}}(\{D(S_R, S_{F,j})\}) \tag{4}$$

where S_R, S_F denote the two specific point sets defined above, $S_{F,j}$ denotes the point sets gathered from the separated pillars of S_F, $D \in \{D_C, D_k\}$ denotes the metric that we use to measure distance between two point sets, $N_{\text{half}} = \lceil |\{S_{F,j}\}|/2 \rceil$ is half of the number of point subsets $S_{F,j}$, and $\text{Top}_k(V)$ denotes the sum of the largest k values in V.

As shown in Fig.3, the local differences of the forged cars are usually larger than the global differences in both chamfer distance and average square L2 distance of kNN (with all p-value less than 1.0×10^{-11} in Kolmogorov-Smirnov tests). We further compare the local difference and global difference for each forged car, and find that if we choose $D_{\text{avg\_local}}$ as the local difference, there are 55.7% forged cars have larger local difference on the chamfer distance metric, and 54.5% forged cars have larger local difference on the average square L2 distance of kNN metric. If we choose $D_{\text{half\_max\_local}}$ as the local difference, 87.4% forged cars have larger local difference on the chamfer distance metric, and 87.5% forged cars have larger local difference on the average square L2 distance of kNN metric. Similar results are observed when we repeat the experiment above on several other real cars randomly sampled from the training set of KITTI.

In summary, the experimental results imply that *the local features do provide the defender with a trace to distinguish*

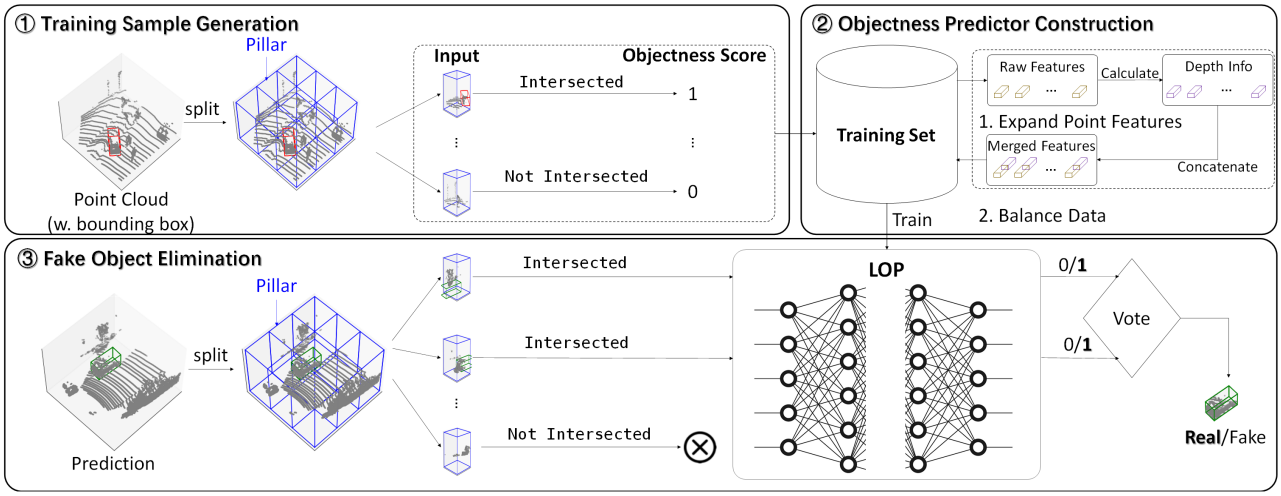

Figure 2: The pipeline of our proposed defense. The input space is split into a number of equal-sized pillars (in the form of blue boxes). The red box in ① represents the bounding box of a ground-truth object during training, while the green box in ③ represents that of a predicted object from the 3D object detector during testing.

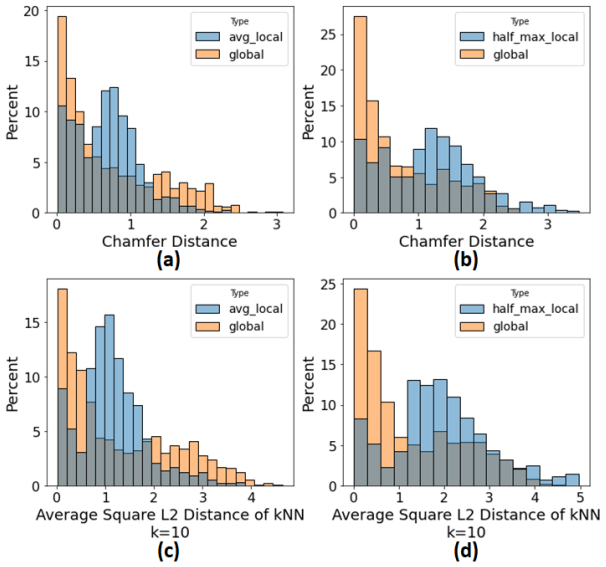

Figure 3: The local and global differences of PCs between real and forged cars (The grey bars inside denote the overlapping region.).

between the real and forged cars. In fact, our insight also conforms to a recent work on enhancing the precision of LiDAR-based object detectors [15], where they suggest that with an appropriate strategy of spatial division, one small part of real objects can also contain rich enough spatial and semantic information to predict the category, bounding box and confidence score of its related object.

4.1.2 Technical Designs

To facilitate the modeling of local object features, in the first stage we prepare a dataset D_{obj} consisting of pairs of points in each pillar from ground-truth objects and an automatically annotated objectness label based on a standard training dataset for LiDAR-based object detectors (e.g., KITTI [19]). Formally, we denote the training dataset as $D = \{(X_t, \{\mathbf{b}_k\}_{k=1}^{N_t})\}_{t=1}^{N}$, where N_t denotes the number of ground-truth objects in the PC X_t, and $\mathbf{b}_k$ denotes the bounding box of the k-th ground-truth object in X_t. First, we split the full $L \times W \times H$ 3D region which covers the input point clouds into a number of pillars $\{p_j\}$ with an equal size $l \times w \times H$, where $l = 1m, w = 1m$ in our implementation. Then for each pillar p_j, we generate an input-output pair, which can be represented as (pc_j, obj_j), as follows:

Generating Input pc_j. We directly collect the inside points of each pillar from the input PC X_t to form the input feature pc_j, i.e., $pc_j = X_t \cap p_j$, composed of a batch of points' features x_i inside p_j. To normalize the generated input, we constrain the size of pc_j as M_{pc}, where M_{pc} is a fixed hyper-parameter. For those pc_j with a larger size, we randomly sample M_{pc} interior points as its input. Otherwise, pc_j is padded with $\vec{0}$ until the size constraint is satisfied.

Generating Label obj_j. We first calculate the 2D Intersection over Union (IoU), the ratio of the area of intersection region over that of union region, between p_j and each ground-truth bounding box $\mathbf{b}_k$ on the x-y plane. For each pillar p_j, we keep the maximal IoU value over all ground-truth bounding boxes. Finally, we compare the maximal IoU value with a fixed threshold T_{IoU}. If this value is greater than T_{IoU}, we annotate $obj_j = 1$ to indicate that the pillar p_j contains a local part of a real object, or $obj_j = 0$ otherwise. Iterating over all the PC

inputs with the pillars, we finish the collection of the training set $D_{\text{obj}} = \{(pc_j, obj_j)\}$. As an analogy to the training task of masked word prediction for pretrained language models [18], this process works in a fully self-supervised manner without any additional information.

4.2 Objectness Predictor Construction

4.2.1 Insight: The Inimitable Depth-Density Law

Meanwhile, we find that, because recent appearing attacks are designed to cause threats in the real world, they are inevitably limited by certain physical constraints imposed by both the attacker's goal and the attack apparatus. As is introduced in Section 3.3, there exist physical upper bounds on the number of added points and the permissible distance between a fake object and LiDAR for recent appearing attacks. Behind these two limitations, we find that the capability of recent appearing attacks is inherently restricted by the *depth-density* law [14]: with existing technology and methods, it is hard to imitate the real-world objects' relation between the *depth*, i.e. the distance between this object and the LiDAR, and the *point density*, i.e. the ratio of the number of input points inside this object's bounding box over the volume of its bounding box.

• **A Pilot Study.** Similar to the reason introduced in Section 4.1, we mainly validate the above observation on cars here. We first randomly sample 1,000 real cars from the training set of KITTI and 1,000 forged cars crafted by the mainstream appearing attacks described in Section 5. Then, we calculate the depth and point density for these objects based on their bounding boxes and the related points. As shown in Fig.4, the point density of real cars is approximately inversely proportional to their depth. In contrast, the point density of the forged cars seems to be independent of the depth: they can have small depth and small point density simultaneously, while this seldom happens for real cars.

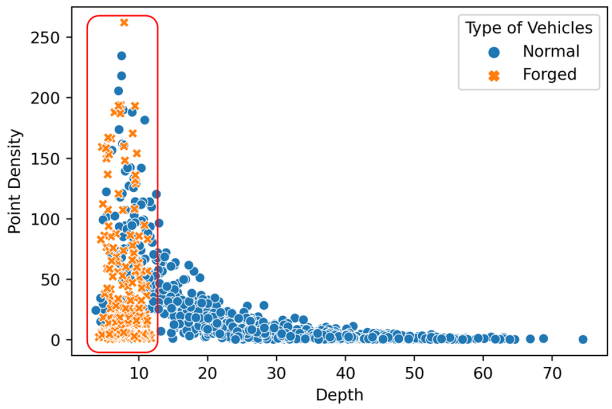

Figure 4: The distribution chart of the depth-density relation. The blue points represent normal cars, the orange crosses represent forged cars, and the red rectangle shows the confounding region of the two.

Though differences exist between real and forged cars in terms of the depth-density relation, it is still hard to directly distinguish them by heuristic algorithms. Due to the complexity of real-world environments, there exists the confounding region in the depth-density relation distribution (highlighted in Fig.4), which is mainly caused by some real cars occluded by others, with smaller depth and point density at the same time. Besides, the complexity is further increased by errors such as the noise in LiDAR perception and the deficiency of attack equipment. In other words, it can be improper to explicitly filter out any detected object based on the hand-crafted rules. As a data-driven approach, we alternatively encourage the LOP to actively learn to model the complicated depth-density relation of real objects, by further incorporating the depth information explicitly into the input feature of each pillar we derive in the first stage.

4.2.2 Technical Designs

At this stage, we augment the input features in our prepared training dataset D_{obj} with the depth information. Specifically, for each generated training sample (pc_j, obj_j) in D_{obj}, we expand the feature of each point in pc_j from an original 4-dim vector $x_i = (x, y, z, int)$ into a 7-dim one $x_i' = (dx, dy, x, y, z, int, dep)$, where (dx, dy) is the point's 2D relative coordinates to the center of its corresponding pillar in the x-y plane, and $dep = \sqrt{x^2 + y^2 + z^2}$ is the point's depth. In our preliminary, we also experimented with an alternative design with no depth information explicit in the input feature. The practice would result in a LOP which is much less effective in distinguishing the forged objects from the real ones than using our current solution.

To adaptively learn the depth-density relation for distinguishing real and forged cars or other obstacles, we implement the LOP O with the architecture of an off-the-shelf backbone PC classifier (e.g., PointNet [34] or DGCNN [45]), considering their validated performance on many downstream 3D tasks. Note that the *negative samples* in D_{obj}, i.e. the generated samples with $obj_j = 0$, are much more than the *positive samples*, i.e. the generated samples with $obj_j = 1$. Thus, we delete a part of negative samples in random to keep data balance and ensure that the ratio of positive samples and negative samples does not exceed 1 : 1.5. To further alleviate the data imbalance problem, we also adopt the idea of focal loss [30] in the learning objective of LOP:

$$FL(p, y) = -\alpha_{fl}(1 - p_y)^{\gamma_{fl}} log(p_y) \tag{5}$$

where the positive constants α_{fl}, γ_{fl} ($\gamma_{fl} > 1$) are the hyper-parameters of the focal loss, which are set by following the best practices in [30]. Besides, p_y is the probability of the y-th class returned by the predictor.

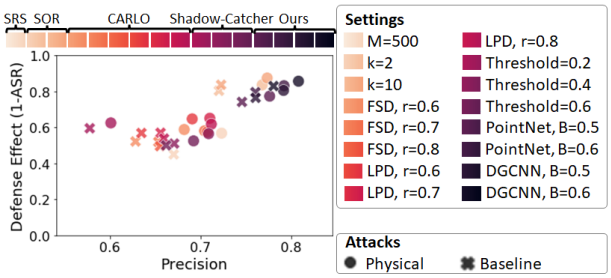

Figure 5: The relation graph of defense effect (1-ASR) and precision on cars of PointPillars under attacks. "PointNet" and "DGCNN" refers to LOP's structure, with a boundary value B used to distinguish real and fake objects as the description in Section 4.3."LPD" and "FSD" are two strategies for CARLO to calculate the anomalous ratio, and M, k, R, Threshold are the hyper-parameters of other defenses, which are all described in Section 3.3.

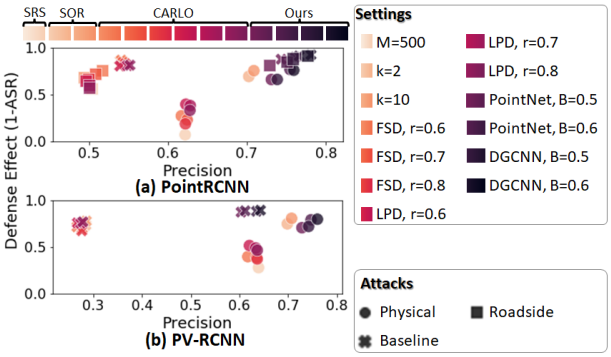

Figure 6: The relation graph of defense effect and precision on cars of (a) PointRCNN and (b) PV-RCNN under attacks.

4.3 Fake Object Elimination

Finally, we leverage the LOP to calculate the objectness score for each pillar intersected with predicted objects, and determine whether these objects are real by a majority voting among the pillars. Specifically, we first divide the detection space into equal-sized pillars, translate the input PC into a series of point subsets inside these pillars and then augment their features, similarly to the former two stage.

Then we use the LOP to calculate a $0/1$ objectness score for each pillar. For each object in the prediction of the 3D object detector, we search for those pillars whose 2D IoU on the x-y plane between itself and the predicted object's bounding box is greater than a specified threshold β, and calculate the sum of their objectness scores as well as the ratio of this sum over the total number of related pillars. Finally, we recognize those objects with the ratio less than or equal to a boundary value B as fake objects, and eliminate them from the prediction.

5 Evaluation and Analysis

5.1 Overview of Evaluation

(1) Victim Models. We choose three mainstream LiDAR-based object detectors: **PointPillars** [27], **PointRCNN** [38] and **PV-RCNN** [37] as the victim models. Specifically, we adopt the implementation of these three object detectors available on an open-source project OpenPCDet [43], each of which is normally trained on KITTI dataset [19] to achieve the near state-of-the-art performance.

(2) Attack Methods. We implement three popular appearing attacks which can be roughly categorized into *white-box* and *black-box* attacks. In the former case, the attacker has full access to the victim 3D detector, including the architecture and the parameters, while the latter only accesses the detector as a black-box prediction API. Specifically, the attacks are

• *A Variant of Adv-LiDAR* [11] (*abbrev.* **Baseline**, *white-box*): Because Adv-LiDAR is specially designed for attacking Baidu's Apollo [1], it could hardly be directly transferred to attack other 3D object detectors [41]. Therefore, following its main idea, we implement a variant of ADV-LiDAR by randomly injecting a certain number of points into a specified zone, and using FGSM [20] to increase the confidence scores of those forged cars related to these points.

• *Yang's Work* [48] (*abbrev.* **Roadside**, *white-box*): This attack forges cars by 3D printing a small and specifically-designed object, increasing their confidence scores and category scores of label car, and enlarging their bounding boxes with gradient descent. Since this attack will generate the adversarial points and then turn them into a physical object, we will only deploy the first part for our experiments. In their original work, this attack mainly aims at breaking PointRCNN with a white-box access. We thus also follow the settings in our experiments.

• *Sun's Work* [41] (*abbrev.* **Physical**, *black-box*): This attack forges cars by duplicating real cars, which contains a limited number of points due to either inter-occlusion or intra-occlusion. The PC of the fake car is then transformed to a front-near position of the victim ADS.

(3) Baseline Defenses. We implement **SRS**, **SOR**, **CARLO** and **Shadow-Catcher** which we have introduced in Section 3.3 as the baseline defenses. We do not consider SVF because it relies on retraining the whole 3D object detector itself, and causes much more time and computation cost compared with other baseline defenses as well as ours (Section 5.2.3).

(4) Metrics. We choose three different metrics to evaluate the performance of our proposed defense and other defenses:

• **Precision** measures the proportion of the real objects in the prediction results. In the context of defense, the decrease in precision reflects whether these defenses would harm the original performance of the victim model. Following [27, 37, 38], we first choose a certain threshold C_{conf} for each 3D object detector, and remove those predicted objects with confidence scores less than C_{conf}. Then we calculate the 2D IoU on the

x-y plane between the bounding boxes of each remaining predicted object and the ground-truth objects. A predicted object is true positive, if its maximal IoU value surpasses another certain threshold C_{IoU} and the predicted category is identical with the ground-truth; otherwise, the predicted object is a false positive prediction. For PointPillars and PointRCNN, we set $C_{\text{conf}} = 0.5$, $C_{\text{IoU}} = 0.5$; for PV-RCNN, we set $C_{\text{conf}} = 0.7$, $C_{\text{IoU}} = 0.5$.

• **Average Precision (AP)** is a comprehensive metric over the precision and the recall of the detection results. Specifically, AP is the average value of precision when the recall is larger than different specific values, which can be represented as

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} max_{r' \geq r} \text{Precision@}(\text{Recall} = r') \quad (6)$$

• **Attack Success Rate (ASR)** measures the ratio of the number of forged cars detected by the victim 3D object detector over the total number of attack attempts, which directly reflects the performance of these defenses. A more effective defense should result in a lower ASR.

(5) Implementation of LOP. We choose two off-the-shelf point-wise PC classification architectures, PointNet [34] and DGCNN [45], to instantiate the LOP. For those hyper-parameters of LOP described in Section 4, we set $M_{pc} = 1024$, $T_{\text{IoU}} = 1 \times 10^{-6}$, $\alpha_{fl} = 1$, $\gamma_{fl} = 2$, $\beta = 1 \times 10^{-3}$.

5.2 Comparison with Baselines

5.2.1 Attack Scenarios

First, we evaluate the performance of our defense against recent appearing attacks. We implement three recent appearing attacks to generate adversarial examples against the three mainstream 3D object detectors based on the KITTI's validation set. We evaluate the ASR of these appearing attacks along with the AP and the precision of these detectors under attacks. Besides the forged cars crafted by appearing attacks, there also remains some normal objects in the adversarial examples, which are considered in the AP and precision metrics. Table 1 and Table 2 show the AP of the detectors on cars when equipped with different defenses, and Fig.5 and Fig.6 plots the defense effectiveness (y-axis, in terms of $1 - \text{ASR}$) and the precision on cars (x-axis) of different defenses.

Results & Analysis. As we can see from Table 1, Table 2, Fig.5 and Fig.6, compared with SRS and CARLO, our defense simultaneously achieves higher defense effectiveness and the victim models under guard have higher AP and precision on cars. For example, under recent appearing attacks, the PointRCNN equipped with the LOP keeps AP on cars over 70% and precision on cars over 72%, while AP on cars is always less than 70% and precision on cars is always less than 63% when deploying SRS or CARLO on the PointRCNN. Compared with SOR, although in some cases our defense has slightly lower defense effectiveness (the margin is less than 5%), it always results in higher AP and precision on cars under attacks. Compared with Shadow-Catcher, although in some cases our defense has slightly lower AP on cars, it always results in higher precision on cars and better defense effectiveness under attacks.

From a different perspective, we observe that other defenses only perform well when protecting certain models against specific attack techniques. For example, SOR performs better when protecting PointPillars, while CARLO performs better when defending against *Physical*. In contrast, the LOP performs well independent of the structure of the 3D object detector or the undergoing appearing attack, which implies that our proposed defense is more general than other defenses.

5.2.2 Benign Scenarios

Then, we evaluate the performance of the victim models under guard on clean samples to measure the performance overhead brought by different defenses. Table 3 and Table 4 present the AP and precision of them in the normal circumstances.

Results & Analysis. As Table 3 and Table 4 show, compared with existing defenses, the performance of these detectors has less degradation in the normal cases when equipped with the LOP than with other defenses. For example, the AP and precision of detectors equipped with other defenses both decrease in most cases, while for these 3D object detectors equipped with the LOP, the AP on cars even increases by $0.33 \sim 1.71\%$, and the precision on cars increases by $2.78 \sim 7.12\%$. Although Shadow-Catcher has slightly higher AP on cars than the LOP, considering the defensive advantages of LOP under different appearing attacks, our proposed defense may be more suitable for practical ADS, due to the performance-robustness balance when the detector is equipped with LOP.

We further analyze why our proposed defense would even increase the performance of the victim models on cars in normal cases, while existing defenses would not: (i) The LOP mainly learns the semantic and spatial features of real objects, while other defenses focus on recognizing fake objects. (ii) The bounding boxes of cars are much larger than that of other objects, which means that there are enough samples corresponding to components of cars provided for the LOP to learn. In summary, our proposed defense incurs almost no damage on the normal performance of the victim models and may sometimes even improve the performance due to its finer granularity modeling of the obstacles. In Appendix A, we further experiment with the hyper-parameters of LOP, which validate that the model structure will not affect the performance of LOP.

5.2.3 Overhead Analysis

Next, we evaluate the overhead in the preparation stage. Except for our LOP and SVF, other defense methods do not introduce additional learning modules and therefore no train-

Table 1: The AP on cars of PointPillars with and without LOP or other defense methods under attacks.

	None	SOR		CARLO					
		k=2	k=10	FSD, r=0.6	FSD, r=0.7	FSD, r=0.8	LPD, r=0.6	LPD, r=0.7	LPD, r=0.8
Physical	70.06%	70.43%	70.01%	65.94%	69.60%	69.64%	67.79%	69.57%	69.82%
Baseline	68.57%	68.84%	68.06%	63.49%	67.96%	67.96%	64.37%	67.99%	68.49%
	None	SRS	Shadow-Catcher			Ours			
		M=500	Threshold=0.2	Threshold=0.4	Threshold=0.6	PointNet, B=0.5	PointNet, B=0.6	DGCNN, B=0.5	DGCNN, B=0.6
Physical	70.06%	70.12%	47.72%	75.46%	77.05%	70.92%	70.97%	71.11%	70.39%
Baseline	68.57%	68.55%	57.81%	75.03%	75.95%	69.50%	69.63%	69.72%	68.61%

Table 2: The AP on cars of PointRCNN and PV-RCNN with and without LOP or other defense methods under attacks.

	PointRCNN			PV-RCNN	
	Physical	Baseline	Roadside	Physical	Baseline
w/o. defense	67.92%	65.95%	61.59%	70.11%	66.39%
SRS (M=500)	69.42%	65.44%	62.48%	70.14%	66.27%
SOR (k=2)	72.56%	65.41%	60.84%	71.43%	64.80%
SOR (k=10)	72.63%	65.26%	60.53%	71.28%	64.09%
CARLO(FSD, R=0.6)	67.16%	63.53%	60.16%	67.99%	63.27%
CARLO(FSD, R=0.7)	68.41%	65.29%	60.71%	70.10%	66.07%
CARLO(FSD, R=0.8)	68.15%	65.22%	60.69%	70.15%	65.84%
CARLO(LPD, R=0.6)	68.98%	65.27%	60.53%	69.23%	64.46%
CARLO(LPD, R=0.7)	69.22%	65.97%	61.41%	70.26%	65.71%
CARLO(LPD, R=0.8)	69.15%	66.04%	61.65%	70.35%	66.11%
Ours(PointNet, B=0.5)	73.32%	71.87%	70.82%	71.47%	69.25%
Ours(PointNet, B=0.6)	**73.77%**	72.30%	71.41%	71.86%	69.19%
Ours(DGCNN, B=0.5)	73.07%	72.29%	71.99%	**71.87%**	**69.88%**
Ours(DGCNN, B=0.6)	73.74%	**73.42%**	**72.84%**	71.51%	68.30%

ing is required in the preparation stage. Table 5 compares the time overhead of LOP and SVF during the preparation phase. Table 6 reports the time and the space overhead of the inference phase of each defense. We conduct the experiments with 5 repetitive tests on each case, and report the mean and the standard deviation as the final results.

• **Results & Analysis.** As Table 5 shows, the time overhead of SVF in the preparation phase is much more higher than that of LOP. It is mainly because SVF requires the retraining of the whole 3D object detectors from scratch, while the training task of LOP only involves a PC-based binary classifier, a much easier learning task compared with that of SVF. More importantly, once LOP is trained, it can be combined with different defense targets to provide the defense, while SVF has to retrain each target.

Meanwhile, Table 6 shows, LOP incurs slightly more time and space overheads than most of the statistical defenses, which can be further reduced by some optimization techniques. For example, to simplify the implementation, we split the whole input space into pillars and use LOP to predict their objectness score during the split in this experiment. However, there is not necessary to check all pillars in the real case. We can identify the pillars which not only intersects with the predicted bounding boxes but also contains points, and only predict their objectness scores to reduce the total times of calculations. Furthermore, we can combine parts of these pillars into a batch and uses LOP to predict in a parallel way for further acceleration. In Section 5.4, we follow the optimization mentioned above to deploy LOP in the end-to-end self-driving system and reduce the time overhead caused by LOP to less than 10ms per detection, which has almost no influence on the real-time requirement of the self-driving system.

5.3 Adaptive Attacks

In this part, we evaluate whether our defense is robust against an adaptive attacker who knows the existence of LOP and in the worst case has the access to the structure and the parameters of our LOP. In this almost worst-case threat model, it is possible for the adversary to attempt to bypass our defense during the generation of forged objects. As the *Physical* attack in [41] requires no training stage in its generation, we choose to modify the *Baseline* attack, i.e., the attack in [11], which we refer to as the *Baseline* attack throughout this response letter, into an adaptive attack against our defense. Specifically, we propose to generate the adversarial point cloud by simultaneously optimizing the original appearing attack objective and maximizing the score of the crafted object under LOP. To enhance the performance of the Baseline attack, we further replace the FGSM algorithm by PGD. Table 7 reports the ASR of the adaptive attacks on the three 3D object detectors when LOP is deployed or not, along with the AP and the precision of the detectors on cars under the adaptive attack.

• **Results & Analysis.** As we can see from Table 7, our LOP performs well when defending against the adaptive attacks above. Both the PointNet-based and the DGCNN-based LOP can reduce the ASR of the adaptive attacks by a large margin, while only a slight loss of performance on clean samples is observed. For example, when defending PointPillars, the ASR is reduced from 45% to 12% with the DGCNN-based LOP, while the decrease of AP is by less than 4%. From our perspective, the result may be because the orthogonality between the original attack target and the intention of bypassing LOP, which brings challenges for optimizing two different loss function at the same time. In summary, LOP has certain robustness against even the worst-case adaptive attack where the attack has a full white-box access to the defense module.

Table 3: The AP and precision of PointPillars on cars with different defenses on clean samples.

	None	SOR		CARLO					
		k=2	k=10	FSD, r=0.6	FSD, r=0.7	FSD, r=0.8	LPD, r=0.6	LPD, r=0.7	LPD, r=0.8
AP	72.34%	71.20%	70.58%	67.92%	72.03%	71.95%	69.80%	71.58%	72.02%
Precision	78.99%	78.91%	78.86%	75.06%	77.81%	78.00%	75.77%	77.56%	78.31%

	None	SRS	Shadow-Catcher			Ours			
		M=500	Threshold=0.2	Threshold=0.4	Threshold=0.6	PointNet, B=0.5	PointNet, B=0.6	DGCNN, B=0.5	DGCNN, B=0.6
AP	72.34%	72.33%	50.58%	77.41%	79.47%	72.86%	72.88%	73.63%	72.73%
Precision	78.99%	79.14%	70.25%	77.31%	76.91%	81.77%	82.38%	83.04%	83.90%

Table 4: The AP and precision of PointRCNN and PV-RCNN on cars with different defenses on clean samples.

	PointRCNN		PV-RCNN	
	AP	Precision	AP	Precision
w/o. defense	75.13%	75.04%	73.32%	73.12%
SRS (M=500)	75.52%	74.75%	73.48%	73.23%
SOR (k=2)	74.46%	74.49%	72.52%	72.92%
SOR (k=10)	74.04%	73.88%	72.22%	72.94%
CARLO(LPD, R=0.6)	73.49%	73.10%	71.43%	70.08%
CARLO(LPD, R=0.7)	74.63%	74.35%	73.21%	72.22%
CARLO(LPD, R=0.8)	74.53%	74.32%	73.20%	72.41%
CARLO(FSD, R=0.6)	74.17%	73.07%	72.00%	69.93%
CARLO(FSD, R=0.7)	74.79%	74.38%	72.94%	71.66%
CARLO(FSD, R=0.8)	74.89%	74.87%	73.15%	72.37%
Ours(PointNet, B=0.5)	76.49%	79.29%	73.65%	77.85%
Ours(PointNet, B=0.6)	76.37%	80.03%	73.80%	78.53%
Ours(DGCNN, B=0.5)	76.77%	80.75%	**74.50%**	79.61%
Ours(DGCNN, B=0.6)	**76.84%**	**81.52%**	73.86%	**80.34%**

Table 5: The time overhead of LOP and SVF during the preparation phase. "*" means that the results are from the OpenPCDet, an open source platform of 3D object detectors, which we use the training time of the specific 3D object detector to approximate the re-training time of SVF on the same detector.

Defense	Total Time (h)	Time Per Epoch (s)	Time Per Iter (s)
SVF (PointPillar)	1.2*	54.0*	8.21*
SVF (PointRCNN)	3.0*	135.0*	20.51*
SVF (PV-RCNN)	5.0*	225.0*	34.19*
Ours (PointNet)	0.41	7.30	0.07
Ours (DGCNN)	0.77	13.88	0.14

5.4 System Integration

To evaluate the system-level usefulness of our proposed defense, we implement the PointNet-based LOP in Baidu's Apollo 6.0.0 system in the optimized way described in Section 5.2.3 , and conduct both the modular and the closed-loop control evaluation in two simulation environments in normal driving scenarios and against the *Physical* attack. We release the implementation details in [3].

• **Experimental Settings.** In the experiments, we construct two different scenarios (e.g., Single Lane Road and Borregas Ave) with random traffic in the LGSVL simulator to evaluate LOP's performance in the end-to-end system. Table 8 reports the ASR of the Physical attack on Apollo 6.0.0, together with the precision and the time cost of the 3D object detectors in Apollo's perception module when LOP is deployed or not, and Fig.7 illustrates the detection results in an end-to-end driving test when the system is deployed without or with LOP, and shows a snapshot of the attacking scenario in the experiments.

Table 6: The time and space overhead of LOP and other defenses during the inference phase.

	Time per sample (s)	GPU Mem (MB)	CPU Mem (MB)
None	0.060±0.005	1477	2551
SRS	0.069±0.007	1473	2549
SOR	0.114±0.005	5827	2516
Carlo (LPD)	0.503±0.003	1477	2552
Carlo (FSD)	2.463±0.005	1477	2506
Shadow-Catcher	0.089±0.002	1477	2551
Ours (PointNet)	1.341±0.011	2283	2518
Ours (DGCNN)	1.589±0.013	3747	2506

• **Results & Analysis.** As Table 8 shows, LOP effectively defends against appearing attacks in the end-to-end Apollo 6.0.0, with a slight proportion of time overhead (less than 10ms). As Fig.7(b) shows, the *Physical* attack can successfully fools Apollo's perception module, and remains existent in the Dreamview even after the processing of MOT. This confirms our argument that appearing attacks is easier to be mounted in practical scenarios than disappearing attacks. In the Dreamview view of Fig.7(c), with the help of our LOP, the forged object is eliminated from Apollo's perception during the evaluation (with ASR= 0%), while the real obstacles remain intact in the perception of the ADS. Therefore, the driving trajectory of the ADS with LOP remains normal and safe during the full driving test. Besides, LOP only incurs a 9.12*ms* overhead on the running time of the 3D detection pipeline on average and slightly brings down the FPS from 29.97 to 23.54, which still satisfies the real-time requirement of a physical self-driving system [21].

Moreover, we use the previously forged objects, which can successfully fool the perception module of Apollo for at least one frame, to further test whether they would lead to a potential harsh braking in different traces. Specifically, we measure whether the self-driving vehicle would do sudden braking, which is shown as it decelerating to 0 km/h in less than 1 second, to calculate the *harsh braking rate*, i.e., the ratio of

Table 7: The performance of LOP against adaptive attack. The names behind "w/o defense" denotes the target LOP of attack.

	PointPillars			PointRCNN			PV-RCNN		
	ASR	AP	Precision	ASR	AP	Precision	ASR	AP	Precision
No Attack	/	72.34%	78.99%	/	75.13%	75.04%	/	73.32%	73.12%
w/o. defense (PointNet)	45.61%	68.52%	66.54%	8.78%	66.14%	55.63%	12.58%	66.37%	28.12%
w/o. defense (DGCNN)	44.78%	68.53%	66.65%	8.56%	65.87%	55.13%	12.53%	66.39%	28.18%
Ours (PointNet, B=0.5)	22.17%	69.36%	74.11%	5.14%	71.95%	74.31%	6.42%	69.27%	60.29%
Ours (PointNet, B=0.6)	16.17%	69.55%	75.87%	4.42%	72.33%	76.06%	**5.58%**	66.39%	61.20%
Ours (DGCNN, B=0.5)	17.11%	**69.73%**	76.30%	3.86%	72.31%	76.41%	6.58%	**69.89%**	63.62%
Ours (DGCNN, B=0.6)	**12.53%**	68.56%	**78.02%**	**3.25%**	**73.33%**	**77.85%**	5.83%	68.37%	**64.48%**

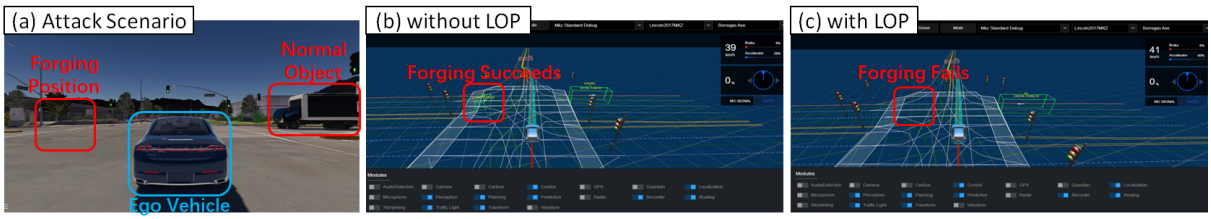

Figure 7: The simulating scenario and the Dreamview of Apollo 6.0.0 without and with our LOP under Physical attack.

Table 8: The performance of the perception module in an end-to-end Apollo 6.0.0 system when the 3D object detector is deployed with or without LOP.

	Precision	ASR	time cost (ms)	FPS
Apollo 6.0.0 (w/o. LOP)	8.33%	53.66%	33.36ms	29.97
Apollo 6.0.0 (w/. LOP)	100.00%	0.00%	42.48ms	23.54

the test cases where the self-driving vehicle suddenly brakes when there is no real obstacle in front of it. We observe that the harsh braking rate of the Apollo without LOP is 13.33% $(2/15)$, while, with LOP, the harsh braking rate is reduced to 0.00% $(0/15)$. We provide the Dreamview snapshot and the details of these experiments in Appendix D. Therefore, combined with the comprehensive evaluation results on the KITTI benchmark, our end-to-end experiments further validate the system-level usefulness of our proposed defense in terms of the improved system robustness, and the acceptable overhead on the running time and the normal driving performance.

6 Discussion

Appearing Attacks vs. Disappearing Attacks. Our current defense mainly focuses on appearing attacks, which form a popular attack class on LiDAR-based object detectors in ADS In contrast to appearing attacks, a disappearing attack aims at hiding the existing objects from the prediction results of the victim 3D object detector [16, 40, 50]. To accomplish this purpose, the adversary would optimally generate a 3D-printing object to the target detector would not recognize it or its neighbouring object, and put the object on the road or near some objects to mount the attack.

In the previous literature, Cao et al. propose one of the earliest disappearing attacks on ADS, and successfully hide the printed objects from the LiDAR-based detection system of Baidu's Apollo by modeling its preprocessing and postprocessing phases into differentiable functions [12]. Later, Tu et al. present a more general disappearing attack which breaks the state-of-the-art 3D object detectors including PointPillars and PointRCNN, and hide the car on which the printed object is positioned from the model's prediction results [44]. Recently, Cao et al. further devise a more powerful disappearing attack, MSF-ADV, which fools the image-based 2D object detectors and LiDAR-based 3D object detectors at the same time, and causes the fusion-based detection system of Baidu's Apollo to ignore the existence of the printed objects [10].

Compared with appearing attack, we argue that a disappearing attack is not physical because it is untargeted and *single-shot*, i.e., the attacker has to put a printed object on the road or near some objects in preparation. This indicates that he/she could hardly choose the victim ADS during the attack. Moreover, the printed object can only take effect once because it might be destroyed or recognized by the people nearby after the first accident happens. In contrast, in an appearing attack the attacker can choose the victim to fire the laser and forge non-existent cars as he/she wishes, making it difficult for others to note the attack due to the almost no evidence left in the accident scene. Nevertheless, considering the severe consequences if happening, how to mitigate disappearing attacks remains a meaningful direction to pursue.

Extension to Other Attack Classes. We further discuss the applicability of our defense for mitigating mis-categorization

attacks, which aims at changing the predicted class of the target objects in the victim's detection results. In this sense, a mis-categorization attack can be seen as the combination of a disappearing attack and an appearing attack. In the above process, we observe that the crafted object would also be left with an abnormal density-depth characteristic which does not belong to the target class. Specifically, in Appendix B, we modify the appearing attacks covered in our experiments into mis-categorization attacks, which selects the objects from the *bicycle* or *pedestrian* classes, and injecting a limited number of points around them to fool the victim 3D detector to mis-categorize them as vehicles, and evaluate the performance of our LOP when deployed alongside the 3D detector. The experimental results in Appendix B show that our proposed defense is also effective against mis-categorization attacks due to the depth-density anomaly introduced by them.

Fusion Models as Defense Targets. We first clarify the relation between our proposed LOP and the fusion models. According to [21], the detection frequency of existing fusion models (including FPN, FCN and AVOD) is usually lower than 15 FPS, and may be unsuitable for real-time self-driving systems due to the efficiency bottleneck. Besides, we suggest our defense is orthogonal to the fusion strategy. LOP in our defense provides a different view for the detectors to confirm their detection, while the fusion strategy incorporates new input modality to enhance robustness. Therefore, instead of viewing fusion models as a comparison group to our defense, we prefer to view the fusion models, which are by essence detectors, as our defense targets. In Appendix C, we provide a preliminary study which validates that our LOP substantially improves the robustness of fusion models against appearing attacks. For example, the PointNet-based LOP would reduce the ASR of the *Physical* attack on EPNet [25] to 0%. In other words, we prefer not to view LOP as a competitor for the fusion models. Instead, LOP empirically improves the robustness of the fusion models, while, as no modifications is made on the image input branch, LOP would not hurt the benefits of fusion models in self-driving systems. For future works, it would be meaningful to systematically evaluate our proposed defense on more representative fusion and 3D object detection models.

Limitation and Future Directions. Finally, we discuss the potential limitations of our proposed defense: According to the case study on the false positives from our defense, we find that our LOP may not recognize the forged obstacles well in some cases due to its uncertainty on distant vehicles. However, due to the existence of the MOT module, the self-driving system keeps refreshing the driving plan and corrects the misprediction of distant objects when the obstacle comes nearby. Moreover, MOT would prevent the self-driving system from ignoring a distant object only if LOP misses a distant object in several consecutive frames, the possibility of which is less than 0.1% according to our calculation. Therefore, the negative influence of LOP on the normal performance of the detector would hardly influence the normal driving behaviors of the defense target. The similar results are also provided in our end-to-end experiments in Section 5.4.

Besides, due to our limited computing resources, we mainly prove the advantages of LOP in terms of computational overhead compared with SVF, while we admit the additional overhead may trade for better defense effectiveness and would not be a problem for most autonomous driving companies. Nevertheless, SVF as a retraining-based approach lies in a different defense category from our proposed plug-and-play defense module. A 3D object detection module which is enhanced by SVF can be further combined with our LOP for better defense effectiveness. As SVF still has a space for improvement in defense effectiveness [41], it would be meaningful for future works to explore their combination in the future.

7 Conclusion

In this paper, we systematically analyze the working mechanisms of recent appearing attacks and summarize their common weaknesses in violating the depth-density law and failing to imitate the local parts of real objects. Based on the defensive insights, we propose a novel plug-and-play defense method which adopts a LOP module to work by side of an arbitrary LiDAR-based object detector to detect and eliminate forged obstacles from its prediction results. To handle the complexity of the depth-density law and the local object feature, we build the LOP with an off-the-shelf point-wise PC classifier and explicitly expand the input point feature with the derived depth information. We present extensive experiments spanning three state-of-the-art 3D object detectors and three known appearing attacks on the standard benchmark KITTI dataset, which validate the effectiveness and flexibility of our proposed defense. Furthermore, we deploy and evaluate the LOP in an end-to-end self-driving system, which validates the system-level usefulness of our proposed defense.

Acknowledgments

We would like to thank the anonymous reviewers and the shepherd for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200), National Natural Science Foundation of China (61972099, U1736208, U1836210, U1836213, 62172104, 62172105, 61902374, 62102093, 62102091). Min Yang is a faculty of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China. Mi Zhang and Min Yang are the corresponding authors.

References

[1] Apollo Open Platform. https://apollo.auto/developer.html. Accessed: 2022-01-30.

[2] ApolloAuto/Apollo. https://github.com/ApolloAuto/apollo/tree/master.

[3] Apollo_LOP. https://anonymous.4open.science/r/Apollo_LOP-A1F4.

[4] Baidu Autonomous Driving Development Kit (Apollo D-KIT). https://apollo.auto/apollo_d_kit.html. Accessed: 2022-01-30.

[5] Combine Lidar and Cameras for 3D object detection - Waymo. https://www.louisbouchard.ai/waymo-lidar/.

[6] Waymo One - Waymo. https://waymo.com/waymo-one/. Accessed: 2022-01-30.

[7] Light detection and ranging. In Shashi Shekhar, Hui Xiong, and Xun Zhou, editors, *Encyclopedia of GIS*, page 1119. Springer, 2017.

[8] Marco Allodi, Alberto Broggi, Domenico Giaquinto, Marco Patander, and Antonio Prioletti. Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle. In *2016 IEEE Intelligent Vehicles Symposium, IV 2016, Gotenburg, Sweden, June 19-22, 2016*, pages 648–653. IEEE, 2016.

[9] Pierre Biasutti, Vincent Lepetit, Jean-François Aujol, Mathieu Brédif, and Aurélie Bugeau. Lu-net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 942–950. IEEE, 2019.

[10] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 176–194. IEEE, 2021.

[11] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2267–2281. ACM, 2019.

[12] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *CoRR*, abs/1907.05418, 2019.

[13] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[14] Jamie Carter, Keil Schmid, Kirk Waters, Lindy Betzhold, Brian Hadley, Rebecca Mataosky, and Jennifer Halleran. Lidar 101: An introduction to lidar technology, data, and applications. *National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center*, 2012.

[15] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan L. Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXI*, volume 12366, pages 68–84. Springer, 2020.

[16] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng (Polo) Chau. Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, volume 11051 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2018.

[17] Hsu-Kuang Chiu, Jie Li, Rares Ambrus, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 14227–14233. IEEE, 2021.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI,*

USA, June 16-21, 2012, pages 3354–3361. IEEE Computer Society, 2012.

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[21] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4338–4364, 2021.

[22] R Spencer Hallyburton, Yupei Liu, Yulong Cao, Z Morley Mao, and Miroslav Pajic. Security analysis of camera-lidar fusion against black-box attacks on autonomous vehicles. In *31st USENIX Security Symposium (USENIX SECURITY)*, 2022.

[23] Abdullah Hamdi, Sara Rojas, Ali K. Thabet, and Bernard Ghanem. Advpc: Transferable adversarial perturbations on 3d point clouds. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XII*, volume 12357 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2020.

[24] Zhongyuan Hau, Soteris Demetriou, Luis Muñoz-González, and Emil C. Lupu. Shadow-catcher: Looking into shadows to detect ghost objects in autonomous vehicle 3d sensing. In *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part I*, volume 12972 of *Lecture Notes in Computer Science*, pages 691–711. Springer, 2021.

[25] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. July 2020.

[26] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 495–502. Omnipress, 2010.

[27] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12697–12705. Computer Vision Foundation / IEEE, 2019.

[28] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989.

[29] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 828–838, 2018.

[30] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007. IEEE Computer Society, 2017.

[31] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artif. Intell.*, 293:103448, 2021.

[32] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12677–12686. Computer Vision Foundation / IEEE, 2019.

[33] Sina Mohseni, Mandar Pitale, Vasu Singh, and Zhangyang Wang. Practical solutions for machine learning safety in autonomous vehicles. In *Proceedings of the Workshop on Artificial Intelligence Safety, co-located with 34th AAAI Conference on Artificial Intelligence, SafeAI@AAAI 2020, New York City, NY, USA, February 7, 2020*, volume 2560 of *CEUR Workshop Proceedings*, pages 162–169. CEUR-WS.org, 2020.

[34] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85. IEEE Computer Society, 2017.

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017.

[36] Yunli Shao, Yuan Zheng, and Zongxuan Sun. Machine learning enabled traffic prediction for speed optimization of connected and autonomous electric vehicles. In *2021 American Control Conference, ACC 2021, New Orleans, LA, USA, May 25-28, 2021*, pages 172–177. IEEE, 2021.

[37] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: point-voxel feature set abstraction for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10526–10535. Computer Vision Foundation / IEEE, 2020.

[38] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 770–779. Computer Vision Foundation / IEEE, 2019.

[39] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 445–467. Springer, 2017.

[40] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*. USENIX Association, 2018.

[41] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 877–894. USENIX Association, 2020.

[42] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[43] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[44] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13713–13722. Computer Vision Foundation / IEEE, 2020.

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019.

[46] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 10359–10366. IEEE, 2020.

[47] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3d adversarial point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 9136–9144. Computer Vision Foundation / IEEE, 2019.

[48] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. Robust roadside physical adversarial attack against deep learning in lidar perception modules. In *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*, pages 349–362. ACM, 2021.

[49] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1198–1207. Computer Vision Foundation / IEEE, 2020.

[50] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1989–2004. ACM, 2019.

[51] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1961–1970. IEEE, 2019.

[52] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4490–4499. Computer Vision Foundation / IEEE Computer Society, 2018.

[53] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11209 of *Lecture Notes in Computer Science*, pages 379–396. Springer, 2018.

A Hyperparameter Sensitivity

Experiment Settings. To explore the impact of the hyperparameters and the structure of LOP, we implement the LOP with different values of B and different architectures. Table 9 report the AP and precision of PointRCNN when our defense is deployed with different settings under the normal circumstance. Fig.8 report the defense effectiveness and the precision of PointRCNN when our defense is deployed with different settings under attacks.

Table 9: The AP and precision of PointRCNN equipped with the LOP with different B and different model structures.

	AP	Precision		
	Car	Car	Pedestrian	Cyclist
w/o. defense	75.13%	75.04%	47.08%	56.87%
Ours(PointNet, B=0.2)	75.96%	76.49%	50.70%	58.66%
Ours(PointNet, B=0.3)	76.21%	77.15%	**50.94%**	58.93%
Ours(PointNet, B=0.4)	**76.50%**	78.05%	50.49%	61.59%
Ours(PointNet, B=0.5)	76.49%	79.29%	49.63%	61.92%
Ours(PointNet, B=0.6)	76.37%	**80.03%**	49.43%	**63.87%**
Ours(DGCNN, B=0.2)	76.44%	77.56%	51.04%	61.03%
Ours(DGCNN, B=0.3)	76.66%	78.36%	**51.31%**	61.72%
Ours(DGCNN, B=0.4)	76.58%	79.44%	51.21%	63.74%
Ours(DGCNN, B=0.5)	76.77%	80.75%	49.59%	65.14%
Ours(DGCNN, B=0.6)	**76.84%**	**81.52%**	49.42%	**67.24%**

Results & Analysis. As we can see from Table 9, the choice of LOP's structure has limited influence on the performance of PointRCNN under the normal circumstance. The differences between their precision are at most 1.49%, 0.72% and 3.37% on cars, pedestrians and cyclists, and the differences between their AP are at most 0.48% on cars. In contrast, the value of B greatly affects the performance of PointRCNN. Normally, the higher value of B is realted with better performance of PointRCNN with the LOP: the precision of PointRCNN on cars and on cyclists increase with a larger B, while the change of AP on cars is always less than 2%.

In fact, the point-wise PC model also performs well in other downstream tasks such as classification and semantic segmentation, which means the key features extracted by them is general enough to handle different CV tasks [34, 35, 45]. Thus, the LOP with different structures can both perform well in recognizing the components of real objects. However, in the pipeline of our proposed defense the value of B directly determines whether a predicted object is preserved or eliminated. Therefore, the value of B affects the performance of 3D object detectors equipped with the LOP.

B Mis-categorization Attack Experiments

• **Experimental Settings.** To implement mis-categorization attack, we follow the idea of **Physical** attack: we collected the PCs about objects which was labeled as pedestrian in the training set of KITTI, and kept the PCs with less than 200 points as the basic data of mis-categorization attack. Then, during each time of mis-categorization attacks, we randomly chose a PC from the basic data and injected it into the target sample, then we further used PGD to change the positions of some points in this PC in order to increased its confidence scores and its classification probability of vehicles. Table 10 reports the ASR of this mis-categorization attack on 3 different object detectors, and the performance of these 3 object detectors with and without our LOP.

• **Results & Analysis.** As we can see from Table 10, LOP reduces the ASR of the mis-categorization attack to almost half of its origin in most cases. For example, LOP reduces at least 58.92% of the original ASR on PointRCNN and reduces at least 49.46% of the original ASR on PV-RCNN. An exception is the PointPillars, which seems to be more resilient against mis-categorization attacks and thus the defense effect of LOP is not as clear as the other two cases. In addition, we also notice a similar phenomenon as discussed in Section of our original manuscript, that LOP can slightly increase the performance of the 3D object detectors in some cases. For example, on PointPillars, the AP increase by 1.13%, while the precision increases 7.52% when the detector is deployed with LOP. In summary, the experimental results validate that LOP is also effective against mis-categorization attacks, and incurs almost no overhead on the performance of object detectors.

C Fusion Experiments

• **Experimental Settings.** According to [22], we choose the fusion model EPNet, which can prevent appearing attacks in a certain degree, as the defense target. Based on the official implementation of EPNet, we implement the *Physical* attack against EPNet, and evaluate its performance on KITTI with or without LOP. Table 11 reports the corresponding results.

• **Results & Analysis.** As we can see from Table 11, without our LOP, the physical attacks can still forging objects in EPNet's detection and bring down the precision of it. However, the ASR of physical attack on EPNet can be reduced to less than 3% with the support of our LOP, and the ASR will further be reduced to 0% when we deploy PointNet-based LOP with

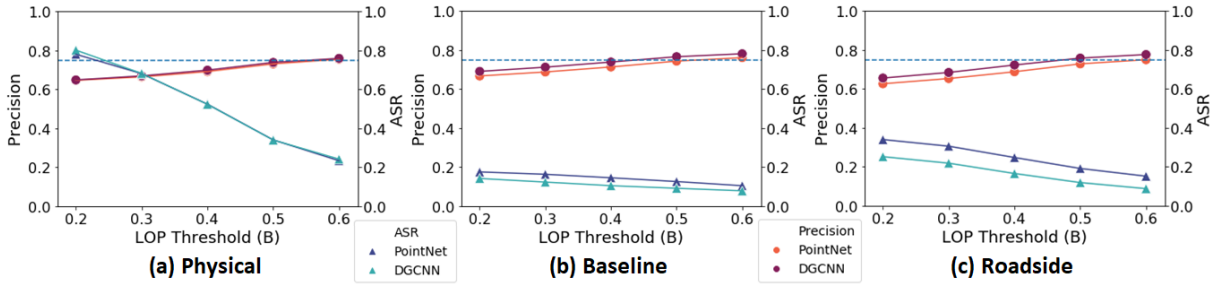

Figure 8: The ASR of different attacks and the precision of PointRCNN when deploying the LOP with different values of B and different model structures on PointRCNN.

Table 10: The ASR of mis-categorizan attacks and the performance of 3D object detectors under this attack.

	PointPillars			PointRCNN			PV-RCNN		
	ASR	AP	Precision	ASR	AP	Precision	ASR	AP	Precision
w/o. defense	4.42%	70.40%	73.78%	13.39%	64.36%	52.22%	12.86%	63.89%	28.51%
Ours (PointNet, B=0.5)	3.67%	70.91%	78.80%	5.50%	72.09%	74.25%	6.44%	67.90%	61.24%
Ours (PointNet, B=0.6)	3.36%	70.96%	79.50%	4.06%	72.85%	76.50%	**5.81%**	68.01%	62.30%
Ours (DGCNN, B=0.5)	3.39%	**71.53%**	80.43%	4.67%	72.50%	75.88%	6.50%	**68.47%**	63.96%
Ours (DGCNN, B=0.6)	**3.06%**	70.18%	**81.30%**	**3.44%**	**73.18%**	**77.83%**	5.89%	67.28%	**65.01%**

Table 11: The ASR of physical attack on EPNet and the performance of EPNet with and without LOP.

	Precision (w/o. attack)	Precision (w. attack)	ASR
None	46.45%	44.96%	44.39%
Ours(PointNet, B=0.4)	61.82%	60.69%	0.53%
Ours(PointNet, B=0.5)	64.19%	63.18%	**0.00%**
Ours(PointNet, B=0.6)	**71.78%**	**69.82%**	**0.00%**
Ours (DGCNN, B=0.4)	59.90%	57.28%	2.67%
Ours (DGCNN, B=0.5)	62.37%	58.33%	0.53%
Ours (DGCNN, B=0.6)	69.23%	67.05%	0.53%

$B = 0.5, 0.6$ on EPNet. Besides, the precision of EPNet with LOP also increased at least 12.32% under attacks and at least 13.45% in the normal circumstances. In summary, we believe LOP should not be viewed a competitor for multi-sensor fusion. Instead, LOP empirically improves the robustness of the fusion models and the 3D object detectors. As LOP has no interruption on the prediction on the image input and simply focus on eliminating malicious objects, LOP would not hurt the benefits of fusion models in self-driving systems.

D Details of The System-level Evaluation

Specifically, to perform these experiments, we first select 5 different fake PCs which are located in front of the self-driving vehicle and can successfully forge the perception module of Apollo or those 3D object detectors for at least 1 frame in the previous experiments. Then, we inject the selected fake PCs into 15 testing traces to conduct the appearing attack against the self-driving vehicle.

We use the Dreamview to visualize the generated future

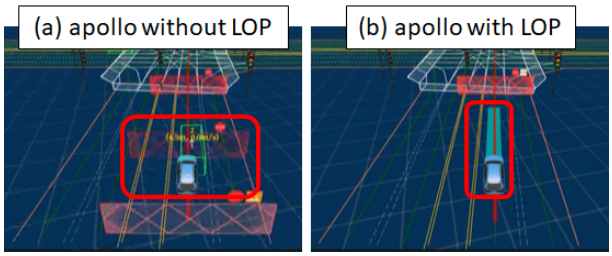

Figure 9: The Dreamview of Apollo without and with our LOP while the attack is aiming at interrupting the route planning.

routes of Apollo 6.0.0, which are shown as the green rectangles in front of the self-driving vehicle and represent the moving trajectories of the self-driving vehicle under the Apollo's control. Based on these routes and the planning module, we define that the self-driving vehicle is harsh braking when the planning module guide the self-driving vehicle decelerate to 0 km/h in less than 1 second.

As shown in the part (a) of Fig.9, the generated future route is extended to the crossroads, which means the self-driving vehicle will move normally and stop at a red light under the instructions of the Apollo with our LOP. Therefore, we consider this situation as "normal". Meanwhile, as shown in the part (b) of Fig.9, the generated future route disappears for a while, and the planning module guide the self-driving vehicle stop in a certain place, which means the self-driving vehicle will falsely brake in the middle of the road. Therefore, we consider this situation as a "harsh braking". We calculate the proportion of the "harsh braking" in the 15 poisoned traces as the harsh braking rate, and report it in Section 5.4.