Robust Roadside Physical Adversarial Attack Against Deep Learning in Lidar Perception Modules

Kaichen Yang
University of Florida
Gainesville, FL, USA
bojanykc@ufl.edu

Tzungyu Tsai
National Tsing Hua University
Hsinchu, Taiwan
s107062519@m107.nthu.edu.tw

Honggang Yu
University of Florida
Gainesville, FL, USA
honggang.yu@ufl.edu

Max Panoff
University of Florida
Gainesville, FL, USA
m.panoff@ufl.edu

Tsung-Yi Ho
National Tsing Hua University
Hsinchu, Taiwan
tyho@cs.nthu.edu.tw

Yier Jin
University of Florida
Gainesville, FL, USA
yier.jin@ece.ufl.edu

ABSTRACT

As Autonomous Vehicles (AVs) mature into viable transportation solutions, mitigating potential vehicle control security risks becomes increasingly important. Perception modules in AVs combine multiple sensors to perceive the surrounding environment. As such, they have been the focus of efforts to exploit the aforementioned risks due to their critical role in controlling autonomous driving technology. Despite extensive and thorough research into the vulnerability of camera-based sensors, vulnerabilities originating from Lidar sensors and their corresponding deep learning models in AVs remain comparatively untouched.

Being aware that small roadside objects can be occasionally incorrectly identified as vehicles through on-board deep learning models, we propose a novel adversarial attack inspired by this phenomenon in both white-box and black-box scenarios. The adversarial attacks proposed in this paper are launched against deep learning models that perform object detection tasks through raw 3D points collected by a Lidar sensor in an autonomous driving scenario. In comparison to existing works, our attack creates not only adversarial point clouds in simulated environments, but also robust adversarial objects that can cause behavioral reactions in state of the art autonomous driving systems. Defense methods are then proposed and evaluated against this type of adversarial objects.

KEYWORDS

Deep learning; security; sensors; autonomous driving

ACM Reference Format:
Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. 2021. Robust Roadside Physical Adversarial Attack Against Deep Learning in Lidar Perception Modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21), June 7–11, 2021, Virtual Event, Hong Kong*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3433210.3453106

ASIA CCS '21, June 7–11, 2021, Virtual Event, Hong Kong

ACM ISBN 978-1-4503-8287-8/21/06…$15.00
https://doi.org/10.1145/3433210.3453106

1 INTRODUCTION

Security concerns on the impact of autonomous driving techniques on road safety have risen along with the rapid development of these techniques. The perception module is a fundamental component in Autonomous Vehicle (AV) systems. It interprets the surrounding environment by combining the capabilities of multiple sensors. The accuracy and efficiency has been significantly improved due to recent advancements in deep learning. Camera sensors and Lidar sensors are most pivotal units in perception systems.

Deep learning models deployed on AVs can help process high-resolution videos captured by the camera for object recognition and tracking in real time. Despite the great success, the information provided by camera-based sensors is still limited to two dimensional (2D) pixels, resulting only rough estimates for the exact distance and location of objects in the environment. To compensate for the limitations of camera-based sensors, Lidar sensors are often applied to provide more accurate three dimensional (3D) information for AV systems. The captured 3D information is presented as coordinates of point cloud data, obtained through firing infra-red (IR) lasers from a Lidar sensor and timing how long they take to return to a corresponding IR detector. Deep learning techniques [16, 33, 35] also help analyze 3D point data captured by Lidar sensors for more comprehensive understanding of the AV's physical environment.

Though deep learning is recognized as a promising solution to plenty of tasks, including perception in an AV system, it remains vulnerable to adversarial examples. Adversarial examples are maliciously crafted inputs and recent works have shown that they can mislead deep learning models [3, 10, 19, 38]. For the camera-based perception based on popular deep learning vision techniques such as Yolo [29] and fast-RCNN [30], many sensor-level attacks leverage this vulnerability benefiting from the abundant research on adversarial examples in computer vision areas [38]. For example, the authors in [19] show that attaching specially designed stickers to traffic signs can cause deep learning models responsible for traffic sign recognition to incorrectly perceive signs. The authors in [10] present a Trojan attack on neural networks trained for AV perception, indicating that adversarial examples can be applied physically against camera sensors with deep learning models deployed in AVs.

Regarding Lidar sensors, however, sensor-level attacks based on the phenomena of adversarial examples is not as straightforward as the one in camera sensors, resulting less discussion on this potential

threat. Existing spoofing or saturating attacks [3, 36] provide the possibility to alter point cloud data captured by Lidar in real road scenarios, yet their effects on the final results of deep learning models are either limited or still not well understood [3]; adversarial attack algorithms [20, 44] have been proposed targeting 3D deep learning models such as PointNets [27, 28] by adding or altering points, but they concentrate only on the point cloud data and may not be implemented in real-world environment.

Recent works [40] start to design physically-realizable attack on the deep learning models in 3D domain, yet their schemes are only verified by desktop 3D scanners, not Lidar sensors on AVs. While the authors in [3] manage to implement the first successful Lidar based perception attack on AVs with physical spoofing restraints, their scheme is only verified in simulation environments. Considering that the laser-spoofing based attack would require using an attack device to track the small target that a Lidar presents on a victim vehicle, which is often moving at high speed, it would be difficult to reliably launch such attacks. The spoofing based attack against Lidar based perception is further extended in [37], where black-box spoofing attacks are launched towards various popular 3D data analysis deep learning models. Despite their success, the authors admit they have yet to evaluate the performance on real AVs. An attack in real road environments has recently emerged [4], but the attack lacks details and sufficient experimental results (neither the code nor the algorithm details are released so far). Others have also attempted to launch adversarial attacks against deep learning models under real Lidar sensors [41], but they again lack real road test results.

To address the limitations of previous attack schemes that fail to simultaneously consider adversarial attacks against deep learning models and Lidar sensors in practical conditions, in this work we propose adversarial attacks against the popular combination of Lidar sensors and deep learning models deployed on AVs. Comparing to the existing attack methods that pay less attention to realize attacks in real road driving scenarios, our attacks are launched with robust physical adversarial objects placed at roadside targeting Lidar sensors with deep learning models that perform 3D object detection tasks in AVs. In the vision of Lidar sensors and deep learning models these small roadside adversarial objects will be recognized as vehicles that invade the lane and cause negative effects including traffic jams, emergency stops or irregular lane changes.

Our attacks include both white-box attacks and black-box attacks. In the white-box attack we assume the adversary has the access to the target deep learning model. In more realistic black-box attacks where deep learning models behind the Lidar may not be accessed by the adversary, we further apply genetic-evolving algorithm to generate adversarial objects. To investigate the potential impact of adversarial objects on traffic, we utilize Lgsvl simulator [31] with Baidu Apollo autonomous driving platform [2] to simulate AV behavior when encountering our roadside adversarial objects. The simulation results illustrate that normal diving behavior controlled by Apollo system will turn into abnormal actions including sudden stop or irregular lane changing when facing the adversarial objects proposed in this paper.

Overall, we design a practical roadside attack in a real world scenario and demonstrate that placing a small roadside adversarial object will cause significant impact on deep learning model without interfering the normal driving behaviors. We also consider the existing defensive methods designed specifically to prevent adversarial 3D point clouds. We further show that our attack can bypass these defense mechanisms with high success rates. Based on our experiments and the understanding of existing defensive mechanisms, we further propose an effective detection method against physical adversarial object by utilizing the physical characteristics of current Lidar sensors in the real road environment.

Overall we make the following contributions in this paper.

- We propose a novel attack to generate adversarial 3D point clouds against deep learning models with high success rates.
- Our attack generates both robust adversarial points in the digital domain and printable adversarial objects that can be implemented in real road environments.
- We show that our attack can bypass existing defenses against adversarial 3D point clouds with high success rates. On the base of that we illustrate possible defense mechanisms that can effectively detect similar physical adversary objects against deep learning models relying on Lidar sensors.
- We also evaluate the impact of our physical roadside adversarial objects on commercial autonomous driving systems. We find that our roadside objects can force an autonomous driving system to perform abnormal decisions including sudden braking and irregular lane changing.

2 BACKGROUND

In this section we will discuss deep learning techniques applied in AV systems and existing adversarial attacks against these deep learning models.

2.1 Deep Learning Based Perception in AV Systems

Camera Based Perception. The progress of deep learning in computer vision for object detection and instance segmentation motivates applications of these deep learning models to perception modules in AV systems. These camera based solutions use images as inputs to extract the locations, types and orientations of objects on the road [43]. However, these methods suffer from the inherent difficulties of estimating depth from images and as a result perform poorly in 3D localization. They are also the targets of various deliberate real world adversarial attacks [19].

Deep Learning on Point Cloud Data. Beyond 2D bounding boxes or pixel masks, 3D understanding of the environment provides more robust information to an AV. With the popularity of 3D sensors increasing, larger amounts of 3D data can be captured and processed to provide precise depth information to mobile devices and autonomous vehicles. Point clouds, a popular data format that contains 3D coordinates information of points sampled from the surface of physical or virtual objects, are widely applied in 3D vision areas such as industrial modeling, surveying, and autonomous driving. Unlike images with ordered pixels, point clouds are unordered, making analysis difficult via popular deep learning techniques. To address this difficulty, Many methods have been proposed to design deep learning models that take point cloud data as inputs and output classification or object detection results. These methods can be

generally divided into three classes: voxel-based methods, bird-eye view methods, and PointNet-based methods.

VoxelNet [49] divides the point cloud into equally-spaced 3D voxels, allowing for ordered analysis. Then 3D Convolutional Neural Networks (CNNs) are applied for 3D bounding box prediction, after which a 2D convolutional detection layer is applied in the final stage. Many recent works [17, 42, 45] adopt this voxel-based architecture and achieve state-of-the-art performance. PointPillar [16], is another example of a voxel-based method. It utilizes an encoder to convert features learned from voxelized point clouds into sparse pseudo images. The final predictions are given by applying 2D CNNs and a SSD-based [21] detection network on the pseudo images. We adopt PointPillar [16] as a target model in the black-box setting as they are supported with strong baseline results and adopted in Baidu Apollo 6.0 system [2].

Using Deep Neural Networks (DNN) to process 2D images is a mature technique and as such, researchers transform Lidar point clouds into ordered 2D structures for 3D object detection in AV systems. These methods [8, 33, 46] convert the point cloud to a bird's eye view representation for efficiency and exploits 2D convolutions. We choose PV-RCNN [33] as one target model in the black-box setting since it achieves high performance in the Kitti bird's eye view benchmark.

Bird-eye and voxel based methods apply deep learning models to analyze point cloud data by decreasing computation cost and improving performance with sparse data. However, these methods cannot directly process raw point cloud data and instead rely on transformation techniques to convert the raw point cloud data into a form that can be easily processed. Though easy to operate, the performance of these schemes is limited by information loss. In comparison, PointNet [27] and PointNet++ [28] apply max-pooling and transformations to reduce the unordered and dimensionally flexible input data to fixed-length global feature vectors. By doing so, they enable end-to-end neural network learning architectures on raw point cloud data. PointNet demonstrates its robustness by introducing the concept of critical points and upper bounds. PointNets are widely adopted in different applications such as 3D object detection [26, 35] as backbone feature generation networks. In this paper, we start our attack against a well-trained Pointnet-based network, PointRCNN [35]. We choose it as the target model in the white-box setting because it achieves high performance in 3D detection test board of KITTI by using only point clouds as the input, and it leads the trend of taking only raw point cloud data as inputs without further pre-processing.

2.2 Adversarial Examples

Szegedy first discovered adversarial examples [38] in which slight but specially chosen alterations, called perturbations, are added to images. With the perturbations, deep learning models may incorrectly classify 2D image inputs despite the fact that human eyes could not detect the changes. Since then, further works [5, 12, 22, 24] have created increasingly complex and effective adversarial examples to models across many domains.

In response to these discoveries, research has identified methods to defend against adversarial examples. Tramer et al. propose adversarial training [39] where a resistant model is created through the addition of known adversarial examples to a training dataset. Defensive distillation [25] smooths adversarial gradients in a model through retraining. This distills the knowledge that a model gains through the first round of training while forcing the adversarial output vectors of the DNN model to converge at a large number. This makes it more difficult for an adversary to trick a model. Guo [13] defends against potentially adversarial inputs with pre-processing methods, such as image compression, transformations, and flips. However, the authors in [5] find possible methods to bypass these defense mechanisms. A further proposed defense to adversarial examples lies in enhancing the robustness of deep learning models. Certifiably robust classifiers whose predictions can be verified to be constant within a given neighborhood may be resistant to adversarial examples [9, 18]. A large amount of work remains to be done to make such a defense practical, however.

Table 1: Comparisons of 3D Adversarial Attacks on Lidar Sensors.

Approach	Practicality in road	Defense discussion	DNN related
[20]	✗	✗	✗
[41]	unverified	✓	✗
[3]	✗	✓	✓
[4]	✓	✗	✓
[37]	✗	✓	✓
Ours	✓	✓	✓

2.3 Lgsvl Simulator and Apollo Platform

Lgsvl simulator [31] is a production-grade Autonomous Driving (AD) simulator based on the Unity 3d engine. It can perform environmental, sensor, and vehicle dynamics and control simulation of a vehicle. Thus, it allow users to customize environments and vehicles for testing and validation. Lgsvl simulator can interface with the Baidu Apollo platform [2], which is an open-source AV system that has over 100 partners and has reached multiple mass production agreements. The simulated vehicle in Lgsvl can be controlled by the Apollo in the virtual environment with perception, prediction, routing and control module. The newest Apollo 6.0 version update its Lidar based perception module based on the PointPillar technique [16].

2.4 Existing Attacks on Lidar Based Perception

Few attacks targeting perception module of AV system consider Lidar sensors and the deep learning module supporting them simultaneously. For the attack against deep learning models, adversarial attacks on 3D point cloud data are proposed in [20, 44]. They demonstrate the feasibility to alter point cloud data for fooling the target deep learning model, but fail to implement these attacks in real world.

Many sensor-level attacks focus on injecting or blocking the point cloud data captured by the Lidar through methods such as laser spoofing or saturation [3, 36]. These types of attacks could trick the victim sensor to provide seemingly legitimate, but actually erroneous, data. However, these attacks lack extension to the deep learning models that give the final results of the detection behind the Lidar sensors, causing the phenomena that the erroneous input

fail to achieve the expected effect on deep learning models [3]. Researchers also explore the adversarial attack against deep learning models in Lidar scenarios [41], yet their work still lacks convincing real road tests. The authors in [3] perform the first study to explore the security of Lidar-based perception in AV settings, but their scheme is only verified in a simulation environment. Their laser-spoofing based attack requires dynamically aiming an attack device at the Lidar on a victim car with high precision, resulting the high difficulty to be piratically launched in real road environment. Though the authors in [37] extend this work into black-box setting, the same issue remains that there lacks practicality in real road environment. The authors in [4] start to launch attacks in real road environment, but their attack lack details, e.g., code or algorithms, and the experimental results are insufficient.

Considering the limitations of current studies of adversarial attacks against Lidar sensors and their corresponding deep learning models in AVs, we propose a novel method that can be launched in a real road environment. We also evaluate the effect of existing defense mechanisms. Based on the evaluation we propose new algorithms to defend against our proposed attack. The major differences between our work and previous studies are shown in Table 1.

3 METHODOLOGY

3.1 Threat Model

Our work assumes an adversary who wishes to attack a deep learning based object detection module deployed on the AV system through its Lidar sensors. By manipulating small roadside objects in the physical world around the target vehicles, the adversary seeks to impede the normal driving behavior of a victim AV. The adversary does this through crafting and placing objects that create illusory vehicles in the victim vehicle's lane. The adversary launches the attacks in both white-box scenarios and black-box scenarios. In the white-box setting the adversary launches the attack against a PointRCNN [35] model trained for object recognition tasks. In the black-box setting the adversary targets PointPillar [16] and PV-RCNN [33] models with only input-output pairs. The attack is conducted by designing and manufacturing certain objects that will be detected by Lidar sensors. The point cloud data generated by the scan results is intended to alter the results of the target deep learning models to achieve the adversary's goals. The adversary generates adversarial watertight meshes and uses a simulated Lidar to confirm that the received point clouds retain their adversarial properties. The attack is then launched in real road environment by placing the 3D printed adversarial object roadside without interfering with human driving behaviors.

3.2 White-box Attacks

3.2.1 Introduction to PointRCNN. Before we start the attacks in white-box scenarios against PointRCNN model, we need to understand how PointRCNN works. PointRCNN is a novel two-stage 3D object detection framework, which directly operates on 3D point clouds and achieves robust and accurate 3D detection performance. The proposed framework consists of two stages, Region Proposal Network (RPN) and Region-based CNN (RCNN). The RPN stage aims at generating 3D bounding box proposals in a bottom-up scheme. The RPN backbone network takes raw point cloud as inputs and outputs classification results and regression locations of each single point. Through a sigmoid layer, the classifications are transformed to probabilities, which determines whether a single point belongs to a certain object. By utilizing 3D bounding boxes to generate ground-truth segmentation masks, the first stage segments foreground points and generates a number of bounding box proposals from the segmented points simultaneously. Such a strategy avoids using a large number of 3D anchor boxes in the whole 3D space as used in previous methods [15, 49] to reduce computation. The following proposal layer takes the RPN outputs as inputs and outputs bounding box proposals as region of interests (ROIs) after filtering out similar or duplicated bounding boxes. In the RCNN stage, after the 3D proposals are generated, a point cloud region pooling operation is used on the learned point representations from RPN stage. Unlike existing 3D methods that directly estimate global box coordinates, these pooled 3D points are transformed to canonical coordinates and combined with pooled point features and the segmentation mask from stage 1 to refine relative coordinates. This strategy fully utilizes all information provided by the robust stage 1 segmentation and proposal sub-network. The final output of the RCNN stage contains both the classification and the regression location of each bounding box proposal. After post-processing (*e.g.* non-maximum suppression) the final results are obtained.

3.2.2 Targeting Deep Object Detection Models. We start our attack process by determining the possible distribution of point clouds that can lead the target model to desired results. Given a point cloud $x \in \mathbb{R}^{n\times 3}$, we add perturbation $\delta \in \mathbb{R}^{n\times 3}$ to mislead the model into incorrect predictions. Here the perturbation δ are treated as the vectors that describe the *direction* and *magnitude* of shifted points. In this work, we focus on appearing attacks, which aim to deceive the perception module into detecting the presence of target objects, *i.e.,* a car in our case, that are not present in the given scenario. To achieve an appearing attack, both the RPN network and RCNN network should be considered simultaneously. Because the final predictions are based on the ROIs, we have to consider both the objectiveness scores from the RPN network and the final detection scores from the RCNN network. During the attack, we only focus on the top m ROIs from both the networks to make the loss function easier to converge. A supporting reason for this is that because of the non-maximum suppression (NMS) process, the number of ROIs will be limited to a certain amount, and many of the similar bounding boxes will be removed. Therefore, the objective function for mis-detection can be formally defined as follows:

$$\begin{aligned} &\underset{\delta}{\text{argmin}}\ \ L_{cls}(x+\delta) \\ &\text{with}\ \ L_{cls}(x) = \sum_{i}^{m} k_i \cdot \left(Z_{bg}(x)_i - Z_{rpn}(x)_i - Z_t(x)_i\right) \\ &k_i = \begin{cases} 1, & \text{if}\ \ Z_t(x)_i < \gamma \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where t denotes the attack target, $Z_{rpn}(\cdot)$ denotes the objectiveness scores predicted by the RPN network. $Z_{bg}(\cdot)$ and $Z_t(\cdot)$ are the logit values of background and target label from RCNN, respectively. We set m to 100 throughout the whole paper as in the experiment we

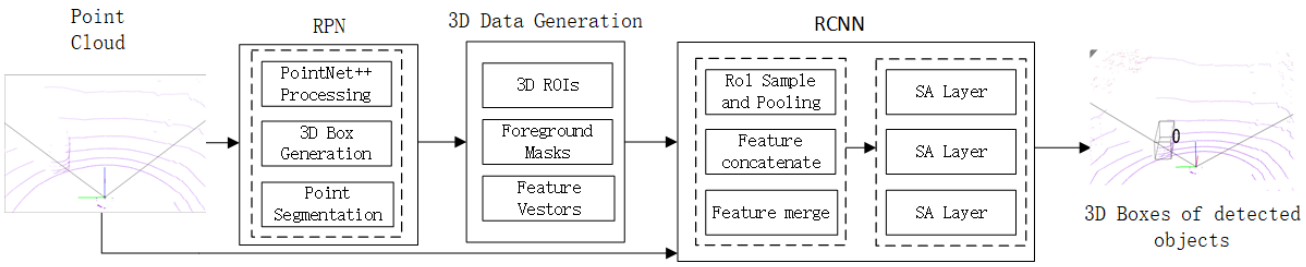


Figure 1: Illustration of the PointRCNN network architecture and workflow.

find it is enough to achieve the desired results; the confidence score γ is set to 0.9.

To simplify converging the objective function, we utilize the concept of Feature Adversary introduced in [32, 47]. They aim to generate adversarial examples by utilizing the features of guide images with the attack target label. More specifically, let x_t be the benign examples with the target label. The adversarial examples $x + \delta$ are generated by solving the following optimization problem:

$$\begin{aligned} \underset{\delta}{\text{argmin}} \;\; & d(\phi_k(x+\delta), \phi_k(x_t)) \\ \text{s.t.} \;\; & d(x+\delta, x) < \tau \end{aligned} \tag{2}$$

where $d(\cdot)$ is the distance function such as L_p norm, $\phi_k(\cdot)$ is the intermediate feature of the k-th layer, and τ controls the amount of the perturbation. This method can effectively generate adversarial examples while targeting different intermediate layers, *i.e.,* different k's. Thus, we apply Feature Adversary on the layer prior to the logits layer and utilize the feature vectors generated by a normal car model. The objective function becomes:

$$\begin{aligned} \underset{\delta}{\text{argmin}} \;\; & L_{cls}(x+\delta) + \alpha \times L_{feat}(x+\delta, x_t) \\ \text{with} \;\; & L_{feat}(x, x_t) = d(\phi_k(x), \phi_k(x_t)) \end{aligned} \tag{3}$$

where $L_{cls}(\cdot)$ is defined in (1), and $\phi_k(\cdot)$ is the layer prior to the layer of classification and bounding box regression in RCNN network.

In addition to the misclassification of nonexistent objects, the bounding boxes of the falsely detected objects are considered to ensure that the objects we generate satisfy the attacker's goals, so that the attack is not rendered ineffective. Specifically, we adjust the orientation of the predicted bounding box to simulate oncoming vehicles, instead of parallel ones. Given the input point set x, the objective function is defined as:

$$L_{box}(x, x_t) = d(\phi_r(x), \phi_r(x_t)) + d(Z_r(x), Z_r(x_t)) \tag{4}$$

where $\phi_r(\cdot)$ denotes the orientation predicted by the RPN, and $Z_r(\cdot)$ denotes the orientation after the bounding box refinement in RCNN. x_t is a clean car model placed at our target location. This function will be considered only if the corresponding objectiveness score is greater than a pre-defined threshold.

3.2.3 Turning Adversarial Points into Physical Objects. We have generated point cloud that can achieve adversarial effects. However, it is still not trivial to transfer these "virtual points" into real road conditions. It is to be noted that though previous works [40] also form adversarial objects, yet the their adversarial objects are generated through surface reconstruction from adversarial point cloud data. The uncertainty in the reconstruction process weaken the attack success rate in real world environment. They also fail to consider the mechanism of specific type of sensors and simply apply random sampling to simulate the 3D sensors. In comparison, We realize our attack by creating physical objects in the beginning of attack that appear as adversarial point clouds when scanned by Lidar. Comparing to the work in [4] who has similar process to generate adversarial objects mainly for hiding purpose, the feature vectors generated by a normal car model enable us to achieve a different goal: deceiving the deep learning models viewing small objects as larger vehicles.

To design such objects, we need to first simulate the working process of Lidar sensors. We build our Lidar simulation tool according to the user manual of Velodyne-Lidar-vlp-16, which is a commercial Lidar sensor using an array of 16 infra-red (IR) lasers paired with IR detectors to measure distances to objects. We also referenced a simulation tool from [14]. By changing the parameters according to the Velodyne manual the simulation of 64-laser Lidar or 128-laser Lidar is also possible. The simulation tool creates a point cloud from a scene based on 3D meshes. Adding a list of models and the scene builder script easily creates random distributions of objects. The Lidar script creates point clouds from these scenes by ray tracing. By setting the parameter according to the configuration, the simulation tool internally creates a set of rays, whose intersections with a given 3D object are then calculated, returning a series of 3D coordinates of point cloud. The difference in the angle between the rays can be specified during initialization of the Lidar class, as well as the position of the Lidar sensor. Specifically, for the Lidar simulation process, the following parameters should be given:

- Lidar parameters, such as position $R = (r_x, r_y, r_z)$, azimuth resolution, etc.
- Object information including vertices $v \in \mathbb{R}^{n\times 3}$ and polyhedrons $p \in \mathbb{N}^{m\times 3}$.

In our adversarial settings, we choose to modify the object vertices v by adding certainly designed perturbations δ, and leave other information untouched. To simulate different viewing angles and distances, we randomly place the Lidar at different pre-defined positions. Given the objects and the Lidar parameters, the simulation process $F(\cdot, \cdot)$ mainly contains the following tasks, which will be described in details in the following sections.

(1) Line-plane intersection: Determine the intersection between the rays and the object surfaces.

(2) Point in polygon: Check whether there is any polygon, *i.e.,* surface, that the point is inside of it.
(3) Distance comparison: Filter out the points belonging to the polygons that are obstructed by other obstacles.

Given a set of polygons (represented by triangles in 3D space) and Lidar rays (represented by lines in 3D space), the points captured by Lidar can be calculated by intersecting the lines with the polygons. Considering a polygon $p_i = \{v_0, v_1, v_2 \mid v_i \in \mathbb{R}^3\}$ and a ray denoted by its origin $R = (r_x, r_y, r_z)$ and direction $D_i = (d_x, d_y, d_z)$, we first let $n = (n_x, n_y, n_z)$ be the normal vector of the polygon p_i. The point $x_i \in \mathbb{R}^3$ can then be given by:

$$x_i = R + D_i \times \frac{(v_0 - R) \cdot n}{D_i \cdot n} \tag{5}$$

Since this process is differentiable, we can integrate this procedure into our optimization problem, so that it can be solved directly using existing mechanisms. In practice, the gradients can be automatically calculated within most of the deep learning frameworks including TensorFlow and PyTorch.

After the line-plane intersection, points that are either obstructed by other obstacles or not located inside the polygons will be filtered out. The remaining points are the input to the PointRCNN.

To summarize, given an object $X = \{v, p\}$ and Lidar sensor parameters S, the input points x captured by the sensor are calculated according to the procedure mentioned earlier. Thus, our physical world attack aims to mislead the model by adjusting the vertices of the object v so that the points captured by sensors will be altered accordingly.

Let $F_S(v, p)$ be the Lidar simulation function that generates point clouds, and $x' = F_S(v + \delta, p)$ be the point clouds captured from our adversarial objects. The objective function is formulated as:

$$\underset{\delta}{\text{argmin}} \; L_{cls}(x') + \alpha \times L_{feat}(x', x_t) + \beta \times L_{box}(x', x_t) \tag{6}$$

3.2.4 Physical Printing Constraints. In the previous section we demonstrate the algorithm can design objects that have specialized shapes such that they are classified as a target class by the victim model. However, to realize these objects into a real road environment, we need to consider many physical constraints. Considering the volume of popular 3D printer, we limit the size of adversarial objects to 45cm × 45cm × 41cm, which is much smaller comparing to common vehicles or pedestrians. We also force a large plane on the bottom of the object so that the object can easily support itself.

Since our target is to generate an object that can mislead the model, the "dis-similarity" between the original one and the modified one is not a critical concern. That is, the object we generate do not have to be similar to the original one in our attack settings. Thus, the distance constraints, such as L_2 norm or Chamfer distance, will not be considered in our objective function to obtain more flexibility in forming the shape of adversarial objects.

Instead of adding another term to objective function, we directly decrease the perturbations to make sure that the length of each axis will not exceed the pre-defined threshold. For example, if the length of the object in x axis is greater than the threshold l_x, the following action will be applied to enforce the size limit.

$$\max\{v_i \mid v_i \in v\} - \min\{v_i \mid v_i \in v\} > l_x \tag{7}$$

To make sure that the object can stand on the ground stably, we also maximize the area of the object that fit on the x-y plane. First, let $p_i = \{v_0, v_1, v_2 \mid v_i \in v\}$ be a plane that fit on x-y plane, *i.e.,* the values of z axis are all zeros. The area can be calculated by:

$$L_{area}(p) = \|(v_1 - v_0) \times (v_2 - v_0)\|_2 \tag{8}$$

Let $p_g \subset p$ be the set containing the polygons on x-y plane, the objective function can be re-written as:

$$L_{area}(v, p) = \sum_{\{v_0, v_1, v_2\} \in p_g} \|(v_1 - v_0) \times (v_2 - v_0)\|_2 \tag{9}$$

where $\times$ is the cross product between two vectors. For those points, only the x and y coordinates will be modified during the optimization progress to make sure that there will be at least one polygon that the object can rest on.

Finally, let the input object be $X = \{v + \delta, p\}$, and the objective function for our attack can be formulated as follows:

$$\begin{aligned} \underset{\delta}{\text{argmin}} \; & L_{cls}(x') + \alpha \times L_{feat}(x', x_t) \\ & + \beta \times L_{box}(x', x_t) + \gamma \times L_{area}(v + \delta, p) \\ \text{with} \; & x' = F_S(v + \delta, p) \end{aligned} \tag{10}$$

where α, β, γ are hyper-parameters that balance the different terms in the objective function, and $L_{cls}(\cdot)$, $L_{feat}(\cdot, \cdot)$, $L_{box}(\cdot, \cdot)$, and $L_{area}(\cdot, \cdot)$ represent classification loss, feature loss, location loss and x-y plane loss. We set $\alpha = 0.001$, $\beta = 0.001$ and $\gamma = -0.001$.

3.3 Black-box Attacks

Unlike white-box attacks, black-box attacks do not require internal information of the target models, but solely rely on input-output pairs. Black-box based attacks can be achieved through several methods. By applying the transferability phenomenon and substitution networks from adversarial attacks, white-box attack models may keep their effectiveness against other models in black-box scenarios [23], but the effects can be unstable. Zeroth Order Optimization (ZOO) [7] launch black-box attacks by modifying the loss function such that it only depends on the output of the DNN, and performing optimization with gradient estimates obtained via finite differences. ZOO, however, suffers from the need for the huge number of queries to the target models.

Our black-box attacks are inspired by work in [6], which first integrates genetic algorithms into black-box adversarial attacks against deep learning models. Genetic algorithm is a population-based gradient-free optimization strategy. It requires well defined genes, populations, fitness functions, mutations and crossovers and recreates nature selection. The population of inputs generated from the genes evolve through mutation and crossover to maximize their fitness score. At every iteration, the candidate with the highest fitness is preserved while the rest are replaced. New candidates are generated by mutating and crossing over a pair of old candidates. In our attack algorithms, the gene and corresponding populations are meshes defined by vertices and faces. The genes are variants of unit icosphere with small Gaussian noise. Mutation is achieved by adding small random perturbation to the population according to a

Table 2: Performance of the target models on car detection.

Model	Easy	Moderate	Hard
PointRCNN	95.92%	91.90%	87.11%
PointPillar	94.82%	91.82%	88.57%
PV-RCNN	98.17 %	94.70%	92.04%

pre-defined mutation chance. Crossover describes the process in which two candidates exchange their elements. The fitness function evaluates the quality of each population member, which encourage the population to evolve to maximize that function.

In our adversarial attack scenario, the fitness function includes the following objectives that overall need to be minimized: A CW like attack objective [5] which motivate the population so they can be classify as certain object type by the target deep learning models:

$$\begin{aligned} l_{cw} &= f(x+\delta) + c \cdot \|\delta\|_p \\ \text{with } f(x) &= \max\{\max_{i \neq y'}\{Z(x)_i\} - Z(x)_{y'}, \kappa\} \end{aligned} \tag{11}$$

where y' denotes the attack target (in our case, the 'vehicle' label), $Z(\cdot)$ is the output of logits layer, and κ the attack confidence. The hyper-parameter denoted as c is used to balance the terms in the objective function. In practice, c can be found effectively using binary search. The perturbation δ can be understood as the vectors that describe the *direction* and *magnitude* of shifted vertices.

Several distance metric which help the mesh stay reasonable shape. We also apply clipping function during mutation so the perturbed vertices will not be too far from the realm, including Laplacian loss l_{lap}, edge loss l_{edge} and normal loss l_{nor} of the evolving meshes. The definition of these three loss can be found in [1]. In all the fitness function is:

$$l = l_{cw} + \omega_1 \cdot l_{lap} + \omega_2 \cdot l_{edge} + \omega_1 \cdot l_{nor} \tag{12}$$

where ω_1, ω_2 and ω_3 are coefficients. In this work we set $\omega_1 = 0.1$, $\omega_2 = 1$ and $\omega_3 = 0.01$. Physical constraints in the white-box setting are similarly applied. The details of the genetic-evolving attack can be found in the Appendix.

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

All experiments were carried out on a server with an Intel E5-2623 v4 2.60GHz CPU with 16GB RAM, Ubuntu 18.04, accelerated by NVIDIA CUDA Framework 10.0 and cuDNN 7.0 with two NVIDIA GeForce RTX 2080Ti GPUs. The adversarial objects are printed by Creality CR-10 Max 3D Printer. The Lidar sensor used for scanning is VLP-16 sensor provided by Velodyne. The outdoor test is carried out on an empty road.

Victim Models. We choose PointRCNN as our target model in the white-box attacks and Pointpillar with PV-RCNN in the black-box setting. All these models are trained using the KITTI dataset [11] with network architectures described in the previous section and hyper-parameters proposed by the authors. We select the model file provided by [16, 33, 34], which are trained on the train split (3712 samples) and evaluated on the validation split (3769 samples) and test split (7518 samples) of the KITTI dataset. The performance on validation set can be found in Table 2.

Table 3: Mis-classification rate of our appearing attack using perturbed polyhedrons.

Object	models	Mis-classification Rate
Adv_white	PointRCNN	798/900 (88.17%)
Adv_black	PointPillar	754/900 (83.7%)
Adv_black	PV-RCNN	790/900 (87.7%)

Simulation Settings. We set the parameters of the Lidar simulation tool according to the configuration of the VLP-16 Lidar. The rotation speed per minute (RPM) is set to 600, and the azimuth resolution is set to 0.2°. The vertical range is set to [-15°, 15°], and the vertical resolution is set to 2°.

Genetic Algorithms Settings. In the genetic algorithm in black box attacks, we initialize the mutation standard deviation at 0.05, mutation probability at 0.2, use a population size of 160, and 1000 queries to compute fitness. The ratio of the leftover is set to 0.5.

Autonomous Driving Platform Setting. In the evaluation of impact on real driving situation we choose the Apollo 6.0 [2] as the autonomous driving platform running with Lgsvl simulator [31]. We run the simulation on the complete Baidu Apollo AD system with all functional modules enabled, i.e., localization, transform, perception, prediction, planning, routing, and control. The adversarial objects as well as a similar-size normal cube will be loaded in a single-lane map in Lgsvl simulator, where the Apollo-controlled vehicle will drive along the lane. Since Apollo 6.0 update its Lidar perception module based on PointPillar [16], the adversarial objects are generated along the method in Section 3. The simulated test vehicle is the Lincoln MKZ with 128-velodyne Lidar sensors, which is also the the Baidu Apollo's reference car. The test is carried in the "singlelaneroad" map provided by the Lgsvl website.

4.2 Attacking Through Simulation of Digital Objects

The appearing attack is launched by designing and placing the objects based on the original polyhedron in the simulated Lidar scanning scene. In this attack, we choose to place the Lidar along a pre-defined position set, instead of moving the object. During the attack, the Lidar will be placed at (x, y, z) where $x \in [-3, 3]$, $y \in [-1, 1]$, $z \in [0.7, 0.8]$, and the object is placed at $(4, -2, 0)$. The rates that our adversarial objects are mis-classified as vehicles are shown in Table 3, with the detection threshold set to 0.3. The object with label "Adv_white" is generated in the white-box setting. The object with label "Adv _black" is generated in the black-box setting. The visualization of our objects, as well as their 3D printed version, can be found in the Appendix. For comparison, we visualize the detection results using a normal car model and our adversarial objects, also shown in the Appendix. Our objects are detected as cars with significantly fewer captured points than from a normal vehicle. This indicates that the model can be fooled by only a few points, compared to benign examples.

4.3 Attacking with 3D Printed Physical Objects

We take various physical constraint into consideration and modify the meshes of adversarial objects so that they can be 3D printed in real world. We start the process with the original polyhedron and

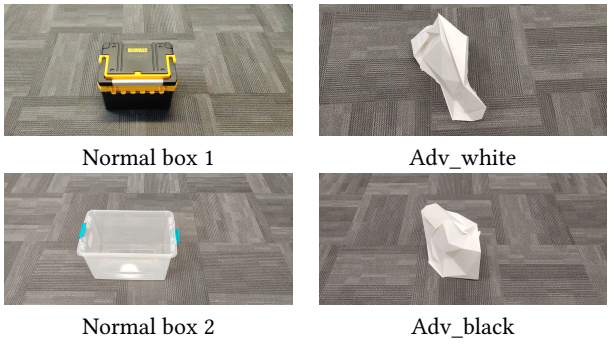


Figure 2: Visualization of the normal box, perturbed object with size limited to $0.45m \times 0.45m \times 0.41m$ used in indoor experiments.

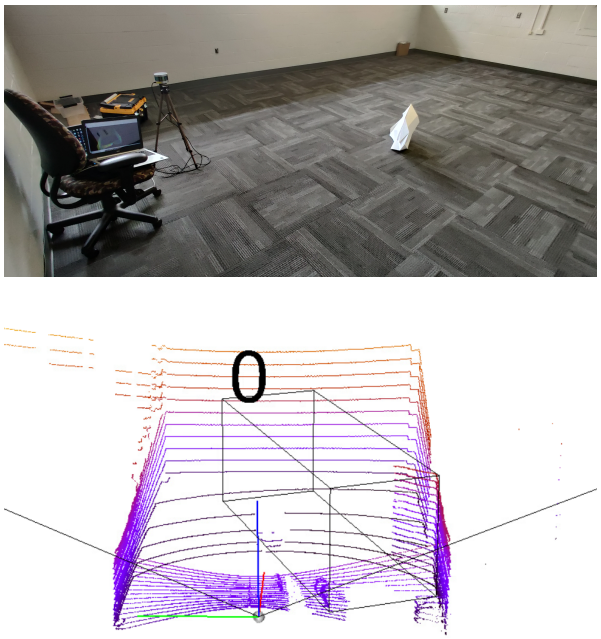


Figure 3: Visualization of appearing attack in indoor scenario with size limited to $0.45m \times 0.45m \times 0.41m$.

"recast" it into an adversarial object that can be captured by Lidar and recognized as vehicle by the victim models. The printed objects are placed in various locations in the front right of the Lidar. We test the adversarial objects in both indoor and outdoor environments. The results are shown in Figures 2 and 4 as well as Tables 4 and 5.

Indoor Experiment. In the indoor environment the height of the Lidar is set to 0.80m and a 0.45m × 0.45m × 0.41m adversarial object is placed across a 4m×7m area in front of the Lidar. We also use two normal boxes with similar sizes as a control. The printed adversarial object and the normal box are shown in Figure 2.

The results of presenting an indoor adversarial object is shown in Figure 3. On the top is a photo showing the environment and on the bottom are the detection results given by the model.

Table 4: Mis-classification rate of normal boxes and adversarial object in indoor scenario.

Object Type	Adv_white	Adv_black	Adv_black
Model type	PointRCNN	PointPillar	Second
Detection Rate	13/15 (86.6%)	14/15 (92.6%)	13/15 (86.6%)
Object Type	Normal 1	Normal 2	Normal 2
Model type	PointRCNN	PointPillar	PV-RCNN
Detection Rate	0/15 (0.0%)	1/15 (6.7%)	1/15 (6.7%)

Figure 4: Outdoor experiment.

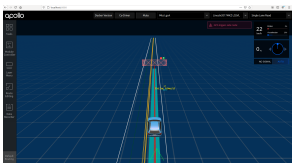
The mis-classification rate demonstrates that a normal box is not recognized as a vehicle by the deep learning models, but our adversarial objects can be continuously recognized as vehicles in different locations of the foreground area (see Table 4). Specifically, Table 4 shows the ratio of number of frames in which the object was detected as vehicle against the total number of frames as well as the mis-classification rate.

Outdoor Experiment. In the outdoor environment the height of Lidar is set to 1.60m and 0.45m × 0.45m × 0.41m size adversarial objects are placed at various roadside locations along the route of the driving vehicle. The distance between the objects and the vehicle is set to the range 1m × 10m. The Lidar is placed on the top of the vehicle. The vehicle will then drive along the road while we record the data from the Lidar. The laptop in the vehicle keep recording the point cloud data captured by the Lidar frame by frame. We again use some normal boxes with similar size as a control. The printed adversarial object and the normal box with vehicle and Lidar are shown in Figure 4.

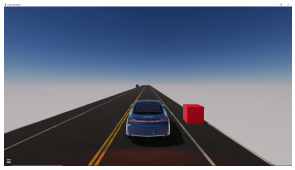
The results from the outdoor point clouds are presented in Table 5. We count the frames that the objects are detected as vehicles to evaluate the attack. From Table 5 we can see that normal box will not be stably recognized as a vehicle by the deep learning model,

Table 5: Mis-classification rate of normal boxes and adversarial object in outdoor scenario.

Object Type	Adv_white	Adv_black	Adv_black
Model type	PointRCNN	PointPillar	PV-RCNN
Detection Rate	213/250 (85.2%)	219/314 (70.0%)	185/240 (77.1%)
Object Type	Normal 1	Normal 2	Normal 2
Model type	PointRCNN	PointPillar	PV-RCNN
Detection Rate	5/220 (2.2%)	3/234 (1.2%)	10/255 (3.9%)



Apollo interface



normal box

Figure 5: Apollo controlled vehicle drive through normal roadside box.

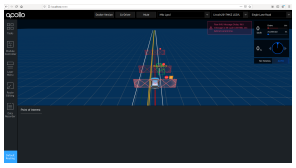
but our adversarial object can be recognized as vehicles in different locations of the foreground area with a high success rate. We also observe that in the vision of Lidar sensor and deep learning models they recognize the small adversarial boxes as a close vehicle, it may cause severe security risks.

4.4 Attack Evaluation on Autonomous Driving

In this section we investigate the reactions of autonomous driving vehicles facing our roadside adversarial objects. A vehicle controlled by Baidu Apollo 6.0 is set to drive on a straight single lane road in the Lgsvl simulator. To start, we place a cube along the road, and observe that the Apollo controlled vehicle does not react and continues its normal driving, as shown in Figure 5. However, when the roadside objects are replaced with adversarial ones, Apollo recognizes it as a extremely close vehicle and blocks the route. Depending on the situation, two actions are taken by the Apollo controlled vehicle. The first action is to stop completely, fully jamming the road. The second reaction is to make a sudden lane change in order to bypass the observed, yet nonexistent, vehicle, which may create a traffic accident as the AV must cross the double yellow line and obstruct traffic flowing the other lane. Figure 5 shows both actions. A video demonstration can be found at ***https://sites.google.com/view/roadsideadversary***.

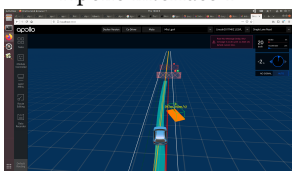
4.5 Performance Comparisons with Similar Works

In this section we provide the attack performance comparisons with similar works in Table 6. Though it may be difficult to compare the success rates directly due to different adversarial settings, the results show that our attack can still obtain comparable success rates while considering physical constraints in real world.

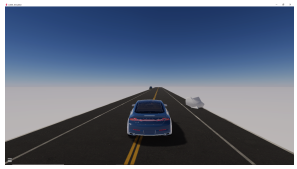


Apollo interface 1

completely stop



Apollo interface 2



lane changing

Figure 6: Apollo controlled vehicle influenced by the adversarial roadside objects.

Table 6: Comparisons of 3D Adversarial Attacks on Lidar Sensors.

Approach	Success rate in simulation	Success rate in road test	Impact on driving system
[41]	80%	No data	No data
[3]	75%	No data	Emergency brake or vehicle freezing
[4]	71%-100%	67%	No data
[37]	80%	No data	No data
Ours	83%-87%	70%-85%	Emergency brake or lane changing

5 EVALUATION OF DEFENSE

In this section, we evaluate our adversarial objects in simulation against existing defense mechanisms, including outlier point removal with kNN distance and the addition of random Gaussian noise proposed in [48]. The experimental results show that these defense mechanisms fail to detect our adversary object both in simulation environment. To effectively detect potential adversarial physical objects that may result in false-positive vehicle detection output, we propose a novel defense method to detect this type of attack based on the physical property of Lidar sensors. Specifically, by measuring the physical relation between the points inside the bounding box and the location of Lidar sensor, we can distinguish points from adversarial objects. We focus on white-box attacks in the defense discussion as it achieve higher success rate in the outdoor experiment, thus reveal more threat.

5.1 Outlier Removal with kNN Distance

This approach is originally proposed in [48] to defend against adversarial attacks designed for PointNet and PointNet++, two widely used 3D point cloud classification models. It is considered that adversarial examples are achieved by shifting points away from the object, so removing distant points should weaken the attack. Thus, this defense aims to find all outlier points and removes them before the point clouds are processed by DNNs. As a quick review, for each input point p, we find its k-nearest neighbors, and calculate

Table 7: Mis-classification rate after kNN outlier removal with $k = 5$ and different α.

α	Mis-classification Rate	Avg. Points Removed
0.0	745/900 (82.8%)	1.17%
0.01	738/900 (82.0%)	1.18%
0.1	723/900 (80.3%)	0.19%

the averaged distance d_p. Then, for all d_p, the mean μ and standard deviation σ are calculated. A point p will be removed if its corresponding d_p satisfies:

$$d_p > \mu + \alpha \times \sigma \tag{13}$$

where α is a user-defined parameter. We evaluate our *Adv_black* adversarial object in simulated scene, as described in previous sections. The results can be found in Table 7. Different α values are used to evaluate the defense, and the averaged percentage of points removed is also reported.

We can found that only a few points will be removed even in different α values (about 1.18%). It means that the average kNN distances of each point cloud are quite similar, so that only a few will exceed the threshold. Due to the low removal rate, this defense does not reduce our success rates significantly.

5.2 Random Gaussian Noise

This defense is implemented by adding random Gaussian noise to the input point clouds. It is considered that the adversarial examples are less stable than the clean ones, since they are likely to locate near the decision boundary to minimize the amount of perturbation. Thus, after adding random noise, the predictions of the clean data should be mostly unchanged. However, the adversarial examples might be affected, and the attack would fail. More specifically, if an input point cloud x is clean, the predictions should be the same:

$$f(x) = f(x + N(0, \sigma^2)) \tag{14}$$

where $f(\cdot)$ denotes the prediction of the model, $N(\mu, \sigma)$ denotes the random Gaussian noise. In contrast, if an input point cloud is adversarial, the predictions might be different:

$$f(x) \neq f(x + N(0, \sigma^2)) \tag{15}$$

Since there are only two classes in our model, which are 'background' and 'car', we directly check whether the object is still detected by the model or not after adding random noise. We evaluate the defense by setting $\mu = 0$ and using 3 different σ^2 values: 0.001, 0.01, and 0.1. The input point clouds are obtained by simulation with our *Adv_black* object. The results can be found in Table 8, with the mis-classification rates using a normal car model for comparison.

As σ^2 increases, the mis-classification rates of both benign and adversarial examples decrease due to the Gaussian noise. However, in some cases, *e.g.*, when $\sigma^2 = 0.01$, the success rates decrease by about 25%. Although some of our adversarial examples are invalidated due to this defense, we still obtain reasonable mis-classification rates.

Table 8: Mis-classification rates with random Gaussian noise added using original car model and Adv_black adversarial object.

σ^2	Car Model	Adversarial Object
0.001	863/900 (95.9%)	745/900 (82.8%)
0.01	849/900 (94.3%)	537/900 (59.7%)
0.1	143/900 (15.9%)	30/900 (3.3%)

Table 9: Accuracy of the SVM for density based defense.

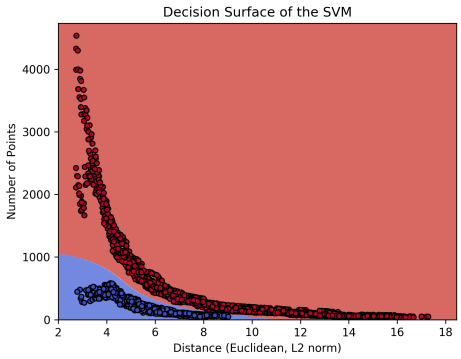
Dataset	Object	Accuracy
Training Set	Car 1	885/900 (98.33%)
	Adversarial object	890/900 (98.89%)
Test Set	Car 2	883/900 (98.11%)
	Adversarial object	893/900 (99.22%)

5.3 Defense based on Physical Properties

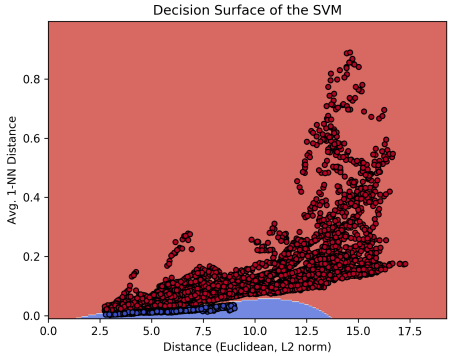
The results in the previous sections reveal that it is nontrivial to distinguish points generated from adversarial objects from those of real objects through their features. However, we observe that the points from the adversarial objects have different physical properties when compared to points from normal vehicles, and these differences could be applied to design detection mechanisms.

Density based Method. From our observation, since the adversarial objects (either natural or artificial) are smaller and lower than normal vehicles in real world due to the size limitation, the points captured by Lidar should be less than the ones from vehicles. Thus, we propose a defense approach based on the number of points within a pre-defined region of each predicted bounding box. This defense aims to distinguish the adversarial bounding boxes from the benign ones. Our defense mechanism takes action after a car is detected by the model, it further determines whether the detection comes from real vehicles or roadside objects. More precisely, for each predicted bounding box, we find the point p_n which is nearest to the Lidar sensor by Euclidean distance and located inside the bounding box. Then, we calculate the number of points that are located inside the sphere with center p_n and radius r. In our settings, we set $r = 0.35$, which is considered large enough to separate the adversarial objects and normal vehicles. To distinguish the benign and adversarial bounding boxes, we simply utilize a Support Vector Machine (SVM) to classify the predicted results. We use a normal car model (denoted as *Car 1*) and our adversarial objects to generate the training set, and we test the SVM using the other adversarial objects and another normal car (denoted as *Car 2*). Both the car models are collected from the Internet, and the visualization can be found in Figure 8. The accuracy of our SVM to distinguish real vehicles and adversarial object can be found in Table 9. The decision boundary of the SVM is shown in Figure 7 (a). We can find that in most cases we can correctly distinguish the adversarial examples from normal vehicles with this property.

Average Distance based Method. From the observation of the point clouds, we find that in some cases, the 1 nearest neighbor (1-NN) distance of a point can be greater than the others. For example, considering the point clouds captured from a vehicle, as shown in Figure 9 (a). Since a normal vehicle is larger than the adversarial



(a) Density based defense.



(b) Average distance based defense.

Figure 7: Visualization of the decision boundary of the SVMs.

Table 10: Accuracy of SVM for average distance based defense.

Dataset	Object	Accuracy
Training Set	Car 1	900/900 (100%)
	Adversarial object	898/900 (99.78%)
Test Set	Car 2	899/900 (99.89%)
	Adversarial object	899/900 (99.89%)

objects, the Lidar may capture the points with various distance from the same object, making the 1-NN distance become larger. In contrast, the points generated from our adversarial objects are located in a smaller region, which can be found in Figure 9 (b) and (c). Thus, we design another defense approach based on the average 1-NN distance of a point cloud belonging to a predicted object: For each bounding box, we calculated the averaged 1-NN distance of the point clouds located inside the box. More specifically, given $\{p_1, p_2, p_3, ..., p_n\} \in P$ be the point sets located inside the bounding box. For each p_i, we find its 1-NN distance by:

$$d_i = \min_{j \neq i} \|p_i - p_j\|_2 \tag{16}$$

The mean 1-NN distance can be obtained by averaging all d_i's. Similar to the previous defense mechanism, we classify the predicted bounding box by a SVM. We use *Car 1* and the adversarial objects to generate the training set, and evaluate the SVM using the *Car 2* and other adversarial objects. The accuracy and the decision boundary of the SVM can be found in Table 10 and Figure 7 (b), respectively. We can find that this defense achieves similar performance, compared to the density based defense. Most of the benign and adversarial points are correctly classified.

6 DISCUSSION AND FUTURE WORKS

6.1 Discussion

Comparison of the Indoor and Outdoor Experimental Results. We observed that the success rate of the attacks decreases significantly when the adversarial object is moved from an indoor environment to an outdoor environment. Several factors may be related to this phenomena. First, the indoor environment is controlled while the outdoor environment is more complex with less regular background points. Furthermore, the uneven floor of the outdoor environment may cause the real point cloud data captured by Lidar deviate from the one we simulated. Another possible reason may be the sparse number of lasers at a low angle our Lidar (VLP-16) has. Once deployed on a high position such as vehicle roof, it may not capture the adversarial object on low ground with enough detail.

Comparison of the white-box and black-box attacks. We observed that white-box attacks and black-box attacks both can generate effective robust adversarial objects in simulation environments and real road environments. Though black-box attacks do not require the internal information of the target models thus can be launched flexibly, the high computation cost lead to far more time to compute an adversarial mesh comparing to the white-box attack. On the other hand, the internal information enhance the attack success rate of white-box attack in both simulated environment and real-world environment.

Origin of the Existence of Adversarial Points. The reasons for the general existence of adversarial examples against many types of deep learning models are still not thoroughly understood. In our case, during the experiment we observe that only 50-100 points from the adversarial objects can trigger the target deep learning model to determine the existence of a vehicle. By viewing the KITTI dataset [11], we find that in two situations small amount of points from normal vehicles are labeled as vehicles: distant vehicles and occluded vehicles. We speculate that the deep learning models trained by the KITTI dataset somehow managed to "remember" the features from these two situations. The adversary points generated from the objects can mimic these features so they can be mis-detected as vehicles by the deep learning models. Similar conclusions are also presented in [37].

6.2 Future Works

Transferability. Though we prove our method effective against PointRCNN models in both digital and physical domains, transferring our method of adversarial object generation to deep learning based perception modules that use other algorithms is still unproven. In subsequent work we will investigate methods to launch

more successful black-box attacks in the scenarios even queries to the target models are more restrictive.

Improving the Robustness in the Real Road Environment. The results in the real road test reveal that our current method to generate adversarial objects neglect some characteristics of the physical world, causing the decreasing of success rate of the attack. In the future, we will adjust the simulation tool and attack process for better performance in the outdoor driving test.

More Defense Mechanisms. In this work we test our adversarial attacks against two simple defense mechanisms specifically designed for potential adversarial 3D points. We also propose two effective defense mechanisms based the different physical property between adversarial points and normal points. However, considering the rapid development of various defense mechanisms against adversarial examples like the recently proposed scheme in Usenix 20 [37], it may be worth effort to explore further defense mechanisms and extend the current attacks to them.

7 CONCLUSION

In this paper, we present a novel attack algorithm that generates manufacturable 3D objects which, when read by Lidar, create adversarial point clouds against deep learning models. This attack starts by searching for purely digital adversarial point clouds. We ensure these point clouds can be created in the physical world by building a watertight mesh from them. We then 3D print this mesh. In our experiments we demonstrate that our attack is feasible both in digital and physical domains. The attack can also be launched in both white-box and black-box scenarios. Our physical objects can function under diverse conditions such as in both road-side tests and indoors. Our method also remains effective even when state of the art 3D adversarial Example defenses are employed. Thus, we additionally propose a detection mechanism effective against this type of attack. The evaluation of the impact on AV control is also presented by importing the adversarial objects into a virtual map. We examine how an Apollo controlled vehicle responds when facing such objects, illustrating the practical risks of placing our adversarial objects along the roadside. We hope our work may provide motivation and inspiration for following works targeting security problems related to Lidar based perception and corresponding deep learning models.

REFERENCES

[1] [n.d.]. https://pytorch3d.readthedocs.io/en/latest/modules/loss.html.

[2] [n.d.]. Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving, howpublished = https://github.com/ApolloAuto/apollo, note = Accessed: 2019-02-11.

[3] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2267–2281.

[4] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. 2019. Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418* (2019).

[5] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the Security and Privacy (S&P) on 2017 IEEE Symposium*. IEEE.

[6] Jinyin Chen, Mengmeng Su, Shijing Shen, Hui Xiong, and Haibin Zheng. 2019. POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm. *Computers & Security* 85 (2019), 89–106.

[7] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 15–26.

[8] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-View 3D Object Detection Network for Autonomous Driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. *CoRR* abs/1902.02918 (2019). arXiv:1902.02918 http://arxiv.org/abs/1902.02918

[10] Shaohua Ding, Yulong Tian, Fengyuan Xu, Qun Li, and Sheng Zhong. 2019. Trojan Attack on Deep Generative Models in Autonomous Driving. In *International Conference on Security and Privacy in Communication Systems*. Springer, 299–318.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[13] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* (2017).

[14] jae251. 2019. lidar_simulation. https://github.com/jae251/lidar_simulation.

[15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. 2018. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1–8.

[16] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12697–12705.

[17] Johannes Lehner, Andreas Mitterecker, Thomas Adler, Markus Hofmarcher, Bernhard Nessler, and Sepp Hochreiter. 2019. Patch Refinement–Localized 3D Object Detection. *arXiv preprint arXiv:1910.04093* (2019).

[18] Alexander Levine, Sahil Singla, and Soheil Feizi. 2019. Certifiably Robust Interpretation in Deep Learning. *CoRR* arXiv:1905.12105 (2019). arXiv:1905.12105 https://arxiv.org/abs/1905.12105

[19] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. 2019. Adversarial camera stickers: A physical camera-based attack on deep learning systems. *CoRR* abs/1904.00759 (2019). arXiv:1904.00759 http://arxiv.org/abs/1904.00759

[20] Daniel Liu, Ronald Yu, and Hao Su. 2019. Extending Adversarial Attacks and Defenses to Deep 3D Point Cloud Classifiers. *arXiv preprint arXiv:1901.03006* (2019).

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

[23] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[24] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Proceedings of the Security and Privacy (S&P) on 2016 IEEE European Symposium*. IEEE.

[25] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (S&P), 2016 IEEE Symposium on*. IEEE, 582–597.

[26] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 918–927.

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 652–660.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[31] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. arXiv:2005.03778 [cs.RO]

[32] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. 2015. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122* (2015).

[33] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *CVPR*.

[34] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2018. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *CoRR* abs/1812.04244 (2018). arXiv:1812.04244 http://arxiv.org/abs/1812.04244

[35] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–779.

[36] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. 2017. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 445–467.

[37] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. 2020. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 877–894.

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[39] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).

[40] Tzungyu Tsai, Kaichen Yang, Tsung-Yi Ho, and Yier Jin. 2020. Robust Adversarial Objects against Deep Learning Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 954–962.

[41] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. 2020. Physically Realizable Adversarial Examples for LiDAR Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13716–13725.

[42] Bei Wang, Jianping An, and Jiayan Cao. 2019. Voxel-FPN: multi-scale voxel feature aggregation in 3D object detection from point clouds. *arXiv preprint arXiv:1907.05286* (2019).

[43] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. 2019. Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[44] Chong Xiang, Charles R Qi, and Bo Li. 2019. Generating 3d adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9136–9144.

[45] Yan Yan, Yuxing Mao, and Bo Li. 2018. Second: Sparsely embedded convolutional detection. *Sensors* 18, 10 (2018), 3337.

[46] Bin Yang, Wenjie Luo, and Raquel Urtasun. 2018. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7652–7660.

[47] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. 2020. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*.

[48] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. 2018. Deflecting 3D Adversarial Point Clouds Through Outlier-Guided Removal. *arXiv preprint arXiv:1812.11017* (2018).

[49] Yin Zhou and Oncel Tuzel. 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A VISUALIZATION OF OBJECTS AND SCENES

A.1 Visualization of the vehicle, objects with printed form

In Figure 8 we demonstrate a few visualizations of car models we use and the resulting adversarial objects created through our method, for both white and black box results. It is clearly evident that both methods produce objects extremely dissimilar to their reference to human senses, but which nonetheless are seen as vehicles by AV perception modules. Note that both objects are easily manufacturable through additive methods, and can independently maintain their correct orientation in the physical world. It is interesting to note that the two shapes are dissimilar from each other in addition to the reference vehicles, with the white box method producing a long angular protrusion that is absent in the black box method.

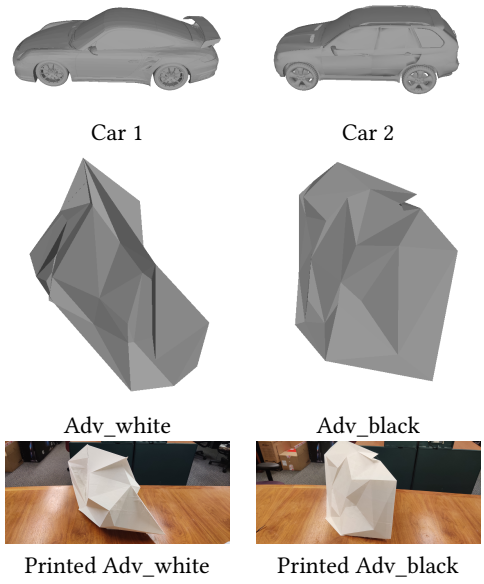
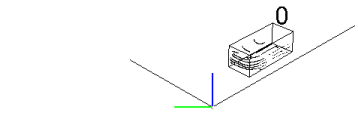


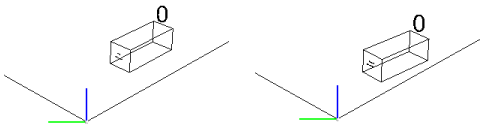
Figure 8: Visualization of car models, our generated adversarial objects and their printed versions.

A.2 Visualization of the scene adversarial objects detected as vehicle

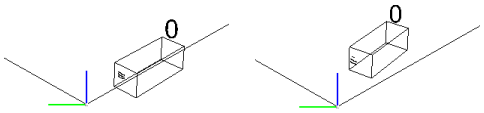
We show in Figure 9 the bounding boxes and point cloud data generated by AV perception systems that detect our adversarial objects. It is evident that although the bounding boxes are appropriately sized, there are far fewer point cloud points in them, and the points are arranged differently than with a typical car model. One can also note that the bounding boxes are always formed perpendicular to a certain plane of point cloud detect ions from the adversarial objects. By controlling the orientation of this plane to a victim, it is possible to control the direction of the resulting bounding box. This also indicates that the generating plane must be clearly visible by the victim for the attack to be successful.



(a) Detection example using normal car model.



(b) Detection examples using Adv_white object.



(c) Detection examples using Adv_black object.

Figure 9: Visualization of detection results using normal car model, Adv_white, and Adv_black objects.

A.3 Genetic-attack algorithm

The pseudocode of our genetic-attack algorithm to generate 3d adversarial objects is shown below in Algorithm 1. This is a fairly standard approach to a genetic algorithm.

Algorithm 1 Genetic-evolving black-box attack

Input: Input mesh x, target model M, number of generations T, population size N, fitness function F, mutate chance c, parameter L.
Output: Adversarial mesh x'

for $population = 1, 2, \ldots, N$ **do**
    Initialize populations by adding small random Gaussian noise to the vertices of x
end for
for $iteration = 1, 2, \ldots, T$ **do**
    Score the populations according to the fitness function F and sort the population according to the scores.
    Keep the best L populations according to the scores as the leftover
    Mutate the the populations in the leftover according to the mutate chance c
    Split the populations in the leftover and crossover them to generate $N - L$ children and append them to the population
end for
return the first population as x'