Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving?

Yi Zhu*[1], Chenglin Miao*[2], Tianhang Zheng[3], Foad Hajiaghajani[1], Lu Su[4], Chunming Qiao[1]
[1] State University of New York at Buffalo, Buffalo, NY USA [2] University of Georgia, Athens, GA USA
[3] University of Toronto, Toronto, ON Canada [4] Purdue University, West Lafayette, IN USA
Email: [1] {yzhu39, foadhaji, qiao}@buffalo.edu, [2] cmiao@uga.edu, [3] th.zheng@mail.utoronto.ca, [4] lusu@purdue.edu

ABSTRACT

As an effective way to acquire accurate information about the driving environment, LiDAR perception has been widely adopted in autonomous driving. The state-of-the-art LiDAR perception systems mainly rely on deep neural networks (DNNs) to achieve good performance. However, DNNs have been demonstrated vulnerable to adversarial attacks. Although there are a few works that study adversarial attacks against LiDAR perception systems, these attacks have some limitations in feasibility, flexibility, and stealthiness when being performed in real-world scenarios. In this paper, we investigate an easier way to perform effective adversarial attacks with high flexibility and good stealthiness against LiDAR perception in autonomous driving. Specifically, we propose a novel attack framework based on which the attacker can identify a few adversarial locations in the physical space. By placing arbitrary objects with reflective surface around these locations, the attacker can easily fool the LiDAR perception systems. Extensive experiments are conducted to evaluate the performance of the proposed attack, and the results show that our proposed attack can achieve more than 90% success rate. In addition, our real-world study demonstrates that the proposed attack can be easily performed using only two commercial drones. To the best of our knowledge, this paper presents the first study on the effect of adversarial locations on LiDAR perception models' behaviors, the first investigation on how to attack LiDAR perception systems using arbitrary objects with reflective surface, and the first attack against LiDAR perception systems using commercial drones in physical world. Potential defense strategies are also discussed to mitigate the proposed attacks.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; • **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Autonomous driving; LiDAR perception; adversarial attack

ACM Reference Format:
Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2021. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving?. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3485377

1 INTRODUCTION

Autonomous vehicles (AVs) are visioned as a revolutionary power for future transportation. Understanding the surrounding driving environment using the sensors such as cameras, radar, and LiDAR, which is called perception, is a fundamental function of AV systems. LiDAR, which can generate point clouds and provide accurate 3D representation of the environment, has drawn significant attention. In recent years, LiDAR is equipped in increasing number of commercial autonomous vehicles [1, 2, 29] to perform perception tasks.

Deep neural networks (DNNs) have been widely adopted in existing LiDAR perception systems in autonomous driving. However, DNNs have been previously found to be vulnerable to *adversarial attacks* where a small perturbation on model's input can significantly change the output predictions. The adversarial attacks have been widely studied in many existing works [10, 19, 28, 32, 49, 66, 68]. There are also some works investigating physically realizable adversarial attacks against image perception systems in autonomous driving [27, 64], which may cause potential traffic accidents in real world scenarios.

Despite the prevalence of adversarial attacks, the study on the vulnerability of LiDAR perception systems to such attacks in physical world remains scant. These attacks can be divided into two categories: laser-based attack and object-based attack. The laser-based attack methods [7, 23, 45] propose to spoof the LiDAR sensor using laser signals to fool the detection systems. Although these methods can achieve good attack performance, the attacker have to dynamically transmit laser signals to the LiDAR with high precision. This makes them difficult to perform in physical world especially when the victim vehicle with LiDAR is moving. For the object-based attack methods, the authors in [3, 8, 50, 51] propose to use objects with adversarial shapes to attack LiDAR perception systems. However, these objects are designed with *specific shapes and sizes*, and attacking with such adversarial objects is not flexible enough in practice. Besides, the abnormal and uncommon shapes of these adversarial objects make them suspicious to human perception. So they can be easily detected by defenders and victims. In addition, it is challenging to generate such specifically shaped objects with high precision in practice.

*The first two authors contribute equally to this work.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea.
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3485377

The above challenges raise an important question: Is there an easier way to perform effective adversarial attacks with high flexibility and good stealthiness against LiDAR perception in autonomous driving? To answer this question, we investigate the possibility of using *arbitrary objects* with reflective surface to fool the LiDAR perception models. Our investigation shows that there are some locations in the physical space where inserting any objects has negative effect on the models' outputs. In this paper, these locations are referred to as *adversarial locations*. If we can derive these locations and place some objects around them, those objects would be able to generate some points around these locations in the point cloud and the LiDAR perception models may be fooled. Here we do not care about the shape and size of the adopted objects. As long as these objects can reflect laser, we can use them to perform the attacks. The intuition behind this attack is that LiDAR perception models learn the geometric features from the locations of the points, thus there are some locations that are important for feature learning. Those inserted points around adversarial locations could distort the geometric features, and may lead to wrong detection results.

Based on the above intuition, in this paper, we propose a new type of attack method based on which the attacker can identify a few *critical adversarial locations* in the physical space. By placing arbitrary objects with reflective surface around these critical locations, the attacker can effectively fool the LiDAR perception model. Our real-world study demonstrates that the attacks can be easily performed using only *two commercial drones*.

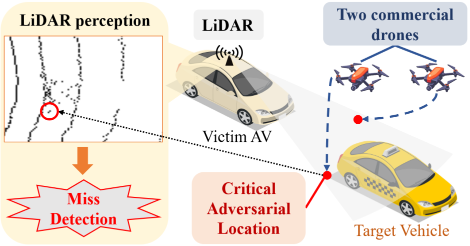


Figure 1: An example of the proposed attack using drones.

Figure 1 shows an example that can be used to illustrate our proposed attack. In this example, the attacker takes two commercial drones as the objects. The attacker first identifies two critical adversarial locations (represented by red dots) in the physical space, and then controls the drones to hover around these locations. Since the drones themselves can reflect the laser and generate points to negatively affect the model's prediction, the LiDAR perception system of the victim AV is fooled and fails to detect the car in front. This example shows that the attack can be easily performed because the attacker only needs to control the drones' locations. In addition, it is difficult for the victim AV to be aware of the attack behavior as the drones just hover for a few seconds and can fly away immediately after the attack without leaving any susceptible objects. Furthermore, after the attack, the drones can be quickly re-deployed to attack another victim AV, which causes serious safety issue on multiple AVs within a short time period. Thus, the proposed attack in this example not only has high feasibility and flexibility, but also can achieve good stealthiness. Certainly, besides drones, we can use other objects, such as traffic signs and advertisement boards, to perform the attacks in practice.

In our proposed attack, a challenging problem is how to derive the least number of critical adversarial locations in the physical space. To address this problem, we propose a novel location search framework to find such locations. This framework contains two steps: *Location Probing* that aims to find a large number of locations with high probability of being adversarial, and *Location Selection* that selects the most critical locations among them based on the negative effect of each individual location. In the Location Selection step, we introduce the concept of *adversarial score* to measure the negative effect of each individual location and propose a selection algorithm to select the least number of critical locations required to achieve the attack goal. The proposed algorithm does not require white box access to the victim perception model, which makes it more practical in real-world scenarios. To make the derivation more efficient, we also design a new algorithm for the Location Probing step to accelerate the process of finding critical adversarial locations.

To demonstrate the effectiveness of the proposed attack, we conduct extensive experiments on both a public LiDAR point cloud dataset and a real-world LiDAR perception testbed. Our experiment results on the public dataset show that the proposed attack can fool the state-of-the-art LiDAR perception model with 93% success rate. We also demonstrate that the proposed attack is robust to different object sizes and location errors generated when placing the object in the physical space. For the real-world LiDAR perception testbed, we perform the proposed attack using only two commercial drones. The experimental results show that the victim AV can be constantly fooled as it drives forward and the attacker can achieve 88.5% success rate.

To the best of our knowledge, this paper presents the first study on the effect of adversarial locations on LiDAR perception models' behaviors, the first investigation on how to attack LiDAR perception systems using arbitrary objects with reflective surface, and the first attack against LiDAR perception systems using commercial drones in physical world. Potential defense strategies are finally discussed to mitigate the proposed attacks.

2 PRELIMINARIES

2.1 LiDAR Perception in Autonomous Driving

LiDAR is one of the major sensors adopted by autonomous vehicles to perform the environment perception. LiDAR generates 3D point cloud that contains the 3D coordinates and intensities of the reflected points through laser scanning. LiDAR perception system is designed to understand the surrounding environment from the collected 3D point cloud, especially to detect the vehicles on the road. Deep learning techniques have made tremendous progress in LiDAR-based object detection. Existing LiDAR object detection models can be divided into three types: voxel-based methods that transform the point cloud into 3D voxel grids and learn the features through 3D CNNs [29, 74], projection-based methods that project the point cloud into 2D feature map and adopt 2D CNNs in feature learning [1, 63], and point-based methods [34, 42] that directly learn the point-wise features through a PointNet-like [35, 36] network.

A common pipeline of those object detection models adopted by LiDAR perception systems is: For the LiDAR object detection model M, it learns the geometric features from the input point cloud D, and outputs a set of bounding box proposals $Y = M(D)$. Each generated bounding box proposal $y \in Y$ contains information of a potential vehicle such as its location, orientation, size, and a confidence score $C(y)$ that represents the possibility of the bounding box containing a vehicle. Those proposals with low confidence scores (smaller than a threshold C_t) will be ignored and the remaining bounding box proposals are the final detection results.

2.2 Adversarial Attacks

Recent studies have proved that deep learning models in many applications are vulnerable to adversarial attacks [4, 5, 15, 31]. By making small perturbations on the original data to generate adversarial examples [9, 20, 33, 65], the attacker can easily fool the deep learning models with high probability.

Given a deep learning model M, input data D and the corresponding ground truth label y^*, the goal of the adversarial attack is to find an adversarial input D_{adv} perturbed from D so that $M(D_{adv}) \neq y^*$. The adversarial input D_{adv} is usually derived by solving an optimization problem [9, 20, 33, 53]. In physically realizable attack, the adversarial input D_{adv} is often generated through a physically realizable modification from the original input D. To attack LiDAR perception models and modify the input LiDAR point cloud in physical world, the easiest way is to insert some points D' into the original point cloud D and get the adversarial input $D_{adv} = D \cup D'$. And an effective approach of inserting points is to place an object with reflective surface and let it be scanned by LiDAR.

3 ADVERSARIAL LOCATIONS

LiDAR perception models learn the geometric features of LiDAR point clouds from the locations of the 3D points. In this paper, we define the *state* of a given location as whether there are LiDAR points around this location. Obviously, the states of different locations in the 3D space determine the inputs of LiDAR perception models and further affect their learned geometric features, which motivates us to perform adversarial attacks by inserting points. Intuitively, changing the states of some locations around the edge of the target vehicle can "distort" the shape/outline of the vehicle, and push the learned geometric features of the target vehicle to the direction that gives low detection confidence. We demonstrate this point in Section 8.2, where we show that the locations around the vehicle's edge are more likely to be adversarial. This is also the reason why inserting points around the edge of the target can make it "disappear". Ideally, if the attacker can change the states of many locations (inserting a large number of points) around the target, it is easy for him to change the learned geometric features and fool the detection results. However, changing the states of so many locations is not practical in the physical world. In our attack, we aim to find a few critical locations that are enough to significantly change the features and fool the detection model. These locations are referred to as *critical adversarial locations*.

We define critical adversarial locations as the locations where inserting any points (even a single point) around each location simultaneously would lead to a failed detection. The reason why there exists critical adversarial locations is explained in Section 8.2. According to the definition, the shape of the inserted points around these critical locations makes no difference in fooling the model. This is because the points around the same location usually have similar contribution to the features of point cloud geometry. To the best of our knowledge, this is the first study that explores the adversarial effects of some specific 3D locations to the target detection model.

4 PROBLEM DEFINITION

Attack scenario. In this paper, we consider the scenarios where the AVs use LiDAR perception system to detect objects on the roads. The goal of the attacker is to fool the perception system and make the victim AVs misunderstand the driving environments. Specifically, we focus on the attack that aims to hide a target vehicle from the LiDAR object detection system, which is called *vehicle hiding attack*. Considering a driving environment where there is a car in front of the victim vehicle, the attacker aims to hide the front car from the LiDAR perception system of the coming victim AV by placing arbitrary objects such as drones around the critical adversarial locations. This kind of attacks may result in a rear-end collision and cause catastrophic consequences.

Threat model. We assume that the attacker can slightly change the targeted driving environments by placing some objects around some specific locations. The attacker cannot obtain the original point clouds collected by the victim AV, but he can generate a series of surrogate point clouds (denoted as D^*) using a similar LiDAR sensor as that adopted by the victim AV. In practice, the attacker can generate the surrogate point clouds by imitating the possible driving behaviors of the victim AV and collecting the sensory data in different directions and from different distances to the target vehicle. In addition, the autonomous vehicle companies may not provide their model details and parameters. Thus, in our attack framework, we consider a black-box setting where the attacker does not have full access to the target model but he can query the model and obtain the outputs.

Problem definition. The problem here is how to find the least number of critical adversarial locations so that the attack goal can be achieved by placing arbitrary objects around these locations. Specifically, we formulate the problem as finding the critical adversarial location set $X = \{x\}, x \in \mathbb{R}^3$ where inserting random point clusters around these locations simultaneously can make the LiDAR perception model fail to detect the target vehicle, while minimizing the number of locations in the set, i.e., $|X|$. Here we use random point clusters to represent the arbitrary objects in the point cloud, and we only care about the output bounding box proposals that are relevant to the target vehicle (e.g., the proposals whose locations are close to the target) among all the output proposals. To measure whether the target vehicle is detected or not by the model, we propose to first select the bounding box proposals that are relevant to the target from all the outputs, and check whether their maximum confidence score is smaller than the detection threshold C_t. Thus, we formulate the problem as the following optimization problem:

$$
\begin{aligned}
\min_{X} \quad & |X| \\
\text{s.t.} \quad & \max_{y \in Y} \epsilon(y) * C(y) < C_t, \\
& Y = M(D^* \cup \{D'(x) | x \in X\}).
\end{aligned}
\tag{1}
$$

Here Y contains all the output bounding box proposals of the detection model M after inserting a random point cluster $D'(x)$ around each location x. And $\epsilon(y)$ checks whether the bounding box proposal y is relevant to the target vehicle y^* and returns 1 if true. A function $F(\cdot)$ is introduced to measure such relevance. If the value $F(y, y^*)$ is larger than a threshold F_t, we can say y is relevant to the target. In this paper, we use *IOU* (Intersection over Union) as the function F, and F_t is set to 0.1.

5 METHODOLOGY

To achieve the attack goal, we need to solve the above optimization problem and derive the least number of critical adversarial locations. However, it is difficult to directly solve the above problem because the objective function is non-differentiable and the constraints are non-convex. To address this challenge, we propose a novel framework based on heuristics to search for the critical adversarial locations. This location search framework contains two steps: *Location Probing* and *Location Selection*. The basic idea of the framework is to first find a large number of locations with high probability of being adversarial, and then select the most critical ones among them by evaluating the negative effect of each individual location on the model's output. This set of the most critical locations is the final output of the location search framework.

Specifically, in the Location Probing step, we aim to find a location set containing a large number of locations that have high probability of being around the adversarial regions and containing the critical adversarial locations. There are many ways to find such location set. We first introduce a vanilla algorithm (in Section 5.1) to achieve this by randomly/uniformly sampling a location set from the given searching area (e.g., around the target vehicle) repeatedly, and selecting the location set that has the largest adversarial impact from all the samples. We also propose a more efficient probing algorithm to find such location set within less searching time, which is introduced in Section 5.3. In the Location Selection step, we aim to select the most critical locations within the found location set. To find out which location is more critical, we define an *adversarial score* to measure the negative effect of each individual location on the model's output. Based on the adversarial score of each location, we then propose a selection algorithm to select the least number of critical locations that can help achieve the attack goal. The Location Selection step is discussed in Section 5.2.

Please note that the proposed framework is general, so different algorithms can be adopted in Location Probing. We propose two algorithms for Location Probing in this paper: a vanilla algorithm based on random sampling (Section 5.1) and an efficient probing algorithm (Section 5.3).

5.1 Location Probing

To find a set of locations with high probability of being adversarial, we need to evaluate the overall adversarial impact of the given location set, i.e., the impact of inserting any points around each location in the set. To measure such adversarial impact, we first define a function $L(X)$ as the maximum confidence score of relevant bounding boxes given the input location set X:

$$L(X) = \max_{y \in Y} \epsilon(y) * C(y), \tag{2}$$

where $Y = M(D^* \cup \{D'(x) | x \in X\})$. Here we randomly insert points around each location x to the point cloud D^*, so $D'(x)$ is a random point cluster whose center is location x. In this step, we could set the number of points in each point cluster (i.e., $|D'(x)|$) as 1 so that we can find the locations where the attacker only needs to insert one point around each of them to achieve the attack goal. This also makes it easy to perform the attack in practice as the attacker only needs to place an object that can generate at least one point around each location. In addition, smaller value of $L(X)$ implies that simultaneously inserting points around the given locations (i.e., X) gives lower confidence score of the proposals and makes the target (i.e., the car in front of the victim AV) harder to be detected. Thus, the locations in the location set X with smaller value of $L(X)$ are more likely to be adversarial.

In this step, we aim to find the location set containing Q locations that have high probability of being adversarial. Based on above discussion, if the value of $L(X)$ is small, most of the locations in X may locate within the adversarial region and has high probability of containing critical adversarial locations. Here we propose a method based on random sampling to find such location set. Specifically, we randomly probe Q locations in the given area (around the target vehicle) for P iterations. In each iteration p, we randomly and independently generate a location set X^p based on uniform distribution, so that we obtain P location sets after P iterations. Among the P location sets, the location set that has the smallest value of $L(\cdot)$ can be selected, denoted as $\hat{X}$. To further improve the probability of the found locations containing critical adversarial locations, instead of selecting one location set, we could select M location sets with the smallest values of $L(X^p)$ among all the P location sets, denoted as $\{\hat{X}^m | i = 1, 2, ..., M\}$. Obviously, larger values of P (number of probing iterations) and Q (number of probed locations at each iteration) make it easier to find the critical adversarial locations with higher probability, due to larger number of probed locations.

The above algorithm is a vanilla version of location probing scheme to demonstrate the basic procedure of the location probing process. In fact, it is possible to employ more sophisticated algorithms. We further propose an *Efficient Probing algorithm* for this step in Section 5.3, which can find $\hat{X}$ with less number of queries and make it easier to find the critical adversarial locations.

5.2 Location Selection

In Location Probing step, we have found M location sets that have high probability of being adversarial. In this step, we intend to remove the redundant locations and select the critical locations with the largest negative effect on the detection model. To help find such critical locations, we first introduce the adversarial score associated with each location.

Adversarial score. As we discussed in Section 2, some locations and regions in the 3D space have negative effect to the model's output. We introduce the concept of adversarial score to measure the negative effect of a given location. The location with high adversarial score is around the region with large negative effect, and inserting points around this location would have large negative effect to the model. To calculate the adversarial score, it is obviously not practical to test all location set combinations and calculate the average effect on the model after inserting points around the location sets. We here propose a method to calculate the adversarial score based on the results in the Location Probing step.

In the Location Probing step, we have probed many sets of locations, and obtained the corresponding values of $L(\cdot)$ which indicate the effect of inserting point clusters around these locations. Based on this, we can use the $L(\cdot)$ value of each location set to measure the effect of inserting new points at these locations. We define a insertion score $W^{insertion}$ for each location $x \in X$ as:

$$W^{insertion}(x) = \exp(L(original) - L(X)), x \in X, \tag{3}$$

where $L(original)$ is the maximum confidence score of relevant bounding box proposals given the original point cloud input D^*. According to the above definition, in the Location Probing step, the insertion score of each location x_q^p in location set X^p can be represented as: $W^{insertion}(x_q^p) = \exp(L(original) - L(X^p))$. Noted that the insertion scores of the locations in the same location set X^p are the same. In the probing process, the probed location set with higher insertion score indicates that the locations in the set are more likely to be adversarial and contain the critical adversarial locations, because inserting points at these locations give lower value of $L(\cdot)$. i.e., have more negative effect to the model's output, as discussed in Section 3.

Since the insertion score only evaluates the overall effect of inserting points around a set of locations together, we still need to evaluate the negative effect of each individual location to find the critical adversarial locations. Thus, a removal score $W^{removal}$ is introduced to evaluate the importance of each location in the given location set. To calculate the removal score, after inserting points around the given location set X, we remove points around a subset of locations that belong to X and then calculate the change of the value of $L(X)$.

Specifically, we randomly remove K locations $\bar{X}^n = \{\bar{x}_k^n | k = 1, 2, ..., K\}$ from X at each iteration n, and calculate the changes of $\exp(L(X - \bar{X}^n) - L(X))$, where $\bar{X}^n \subset X$ and $X - \bar{X}^n$ denotes removing location set $\bar{X}^n$ from set X. After N iterations, the removal score of each location x is given by the average changes:

$$\begin{aligned} &W^{removal}(x) = \\ &AVERAGE(\{\exp(L(X - \bar{X}^n) - L(X)) | x \in \bar{X}^n, n = 1, 2, ..., N\}). \end{aligned} \tag{4}$$

The removal score of the location x measures the importance of this location among all the locations in X. After inserting random point cluster at each location of X, if removing the subset of locations $\bar{X}^n$ gives larger value of $L(\cdot)$, $L(X - \bar{X}^n) > L(X)$), then obviously the removed locations have more negative effect to the model compared with the remaining locations. Thus, the location with higher removal score plays a critical adversarial role to the model's output among the locations in set X.

Finally, we define the adversarial score W of x as the multiplication of the insertion score and removal score:

$$W(x) = W^{insertion}(x) * W^{removal}(x). \tag{5}$$

The reason behind the definition is that higher insertion score indicates location x is in the location set X that has higher probability of being adversarial and being around critical adversarial locations, and higher removal score indicates the location x has larger adversarial effect among the locations in X. The pseudo code of calculating adversarial score of each location x in location set X is shown in Algorithm 1:

Algorithm 1: Calculate adversarial score

```
Input: the surrogate point cloud $D^*$; location set $X$; parameter $N$
       and $K$.
$W^{insertion}(x) \leftarrow \exp(L(original) - L(X)), x \in X$;
for $n \in \{1, 2, ..., N\}$ do
    Randomly remove $K$ locations
      $\bar{X}^n = \{\bar{x}_k^n | k = 1, 2, ..., K\}, \bar{X}^n \subset X$;
end
$W^{removal}(x) \leftarrow AVERAGE(\{\exp(L(X - \bar{X}^n) - L(X)) | x \in$
  $\bar{X}^n, n = 1, 2, ..., N\})$;
$W(x) \leftarrow W^{insertion}(x) * W^{removal}(x)$ for each $x \in X$;
Output: $\{W(x) | x \in X\}$
```

Selection based on adversarial scores. Here we introduce the whole process of Location Selection. Given the input location set $\hat{X}$, we first calculate the adversarial scores $W(x)$ of each location $x \in \hat{X}$ according to Algorithm 1. When calculating the adversarial scores, we can also obtain the values $L(\hat{X} - \bar{X}^n)$ after removing locations $\bar{X}^n$ in each iteration n. We first remove the locations that gives the smallest value of $L(\cdot)$ after removal, $X^{remove} = \arg\min_{\bar{X}^n} L(\hat{X} - \bar{X}^n)$, if removing those locations achieves the attack goal (i.e., $L(\hat{X} - \bar{X}^n) < C_t$). These locations X^{remove} have small negative effect to the model as we discusses before, and removing them in advance would accelerate the process. $\hat{X}$ is then updated to the remaining location set. Starting from an empty set $\ddot{X}$, we then iterative select each location x from the remaining locations in the descending order of their adversarial scores, and check if adding x to the set $\ddot{X}$ makes $L(\cdot)$ decrease, i.e., $L(\ddot{X} + x) < L(\ddot{X})$. If true, we add the location x to $\ddot{X}$ and update $\ddot{X}$. Otherwise, we move on to the next location until we go through all the locations in $\hat{X}$. Then we update the current location set $\hat{X}$ with $\ddot{X}$ if $L(\ddot{X}) < C_t$, and repeat the whole above process until the location set $\hat{X}$ is stable. The algorithm of generating adversarial locations is described in Algorithm 2.

5.3 Efficient Probing Algorithm

Although the above algorithm can effectively generate critical adversarial locations, it requires many searching iterations and queries because it probes the given region uniformly and randomly in the Location Probing step. To accelerate the probing process and further reduce query numbers, we propose Efficient Probing algorithm in replacement of the random sampling algorithm in Section 5.1.

The basic idea of the Efficient Probing algorithm is to utilize the adversarial scores of previous probed locations as guidance. After randomly probing a set of locations, we calculate their adversarial scores. As we discussed in Section 5.2, the location with high adversarial score is around the regions with large negative effect. Then we use the locations with high adversarial scores as the centers, and generate new location set near these locations for the next iteration. In this way, we can always probe the locations around the regions with large negative effect, which increases the probability of finding critical adversarial locations. This probing strategy can narrow the searching space and accelerate the searching process.

Moreover, although we have discussed how to calculate adversarial score in Section 5.2, it brings extra queries. To further reduce the number of queries, we propose a simplified method to estimate the adversarial score approximately. Instead of randomly removing K locations at each iteration, we randomly divide the location set

Algorithm 2: Generate critical adversarial location set

```
Input: the surrogate point cloud $D^*$; parameters $P, Q, M, N$, and $K$.
// Location Probing, a vanilla version
for $p \in \{1, 2, ..., P\}$ do
    randomly probe $Q$ locations $X^p = \{x_q^p | q = 1, 2, ..., Q\}$;
    calculate $L(X^p)$;
end
select top $M$ location sets $\{\hat{X}^m | m = 1, 2, ..., M\}$ among
  $\{X^p | p = 1, 2, ..., P\}$ with the highest values of $L(\cdot)$;
// Location Selection
for each location set $\hat{X}^m$ do
    repeat
        for $n \in \{1, 2, ..., N\}$ do
            Randomly remove $K$ locations
              $\bar{X}^n = \{\bar{x}_k^n | k = 1, 2, ..., K\}, \bar{X}^n \subset \hat{X}^m$;
            Calculate $L(\hat{X}^m - \bar{X}^n)$ after removal;
        end
        // Calculate adversarial scores based on
           Algorithm 1
        $CALCULATE\_ADV\_SCORE(L, \hat{X}^m)$;
        if $\min_{\bar{X}^n} L(\hat{X}^m - \bar{X}^n) < C_t$ then
            // Remove the subset of locations
            $\hat{X}^m \leftarrow \hat{X}^m - \arg\min_{\bar{X}^n} L(\hat{X}^m - \bar{X}^n)$;
        end
        initial $\ddot{X} \leftarrow \emptyset$;
        for $x \in SORT(\hat{X}^m)$ do
            // Iterative select one location and check
               if $L(\cdot)$ decreases
            $\ddot{X} \leftarrow \ddot{X} + x$ if $L(\ddot{X} + x) < L(\ddot{X})$;
        end
        if $L(\ddot{X}) < C_t$ then
            $\hat{X}^m \leftarrow \ddot{X}$;
        end
    until $\hat{X}^m$ is stable;
    get the final $\hat{X}^m$;
end
// Select the shortest location set that achieves the
   attack goal.
$X^* \leftarrow \{X^* | m = 1, 2, ..., M, L(\hat{X}^m) < C_t\}, X^* =$
  $\arg\min_{\hat{X}^m} ||\hat{X}^m||$;
Output: Critical adversarial locations $X^*$
```

X into $\tilde{N}$ subsets $\{\tilde{X}^n | n = 1, 2, ..., \tilde{N}\}$. Then we remove one subset at each time and calculate $L(X - \tilde{X}^n) - L(X)$. The approximated adversarial score of each location $x \in \tilde{X}^n$ is:

$$\begin{aligned}\tilde{W}(x) &= W^{insertion}(x) * \tilde{W}^{removal}(x) \\ &= \exp(L(original) - L(X)) * \exp(L(X - \tilde{X}^n) - L(X)),\end{aligned} \tag{6}$$

where $x \in \tilde{X}^n \subset X$.

In addition, we calculate the adversarial scores and update the searching center only when the probed location set has smaller value of $L(\cdot)$ than previous probed location set, $L(\hat{X}) < L(X^{previous})$. The algorithm of the Efficient Probing is summarized in Section A of the Appendix.

5.4 How to Perform the Attacks in Practice?

In practice, the attacker can derive the critical adversarial locations in an offline manner. Before performing the attacks, the attacker first imitates the driving behaviors of the victim AV and collects the surrogate point clouds D^* in the target driving environment using a similar LiDAR that is adopted by the victim AV. However, the attacker usually cannot predict the behaviors of the victim AV before performing the attacks and it may come from different directions. To address this challenge, we let the attacker collect the surrogate point clouds in different directions and from different distances to the target vehicle. In this way, the attacker can perform robust continue attack and fool the victim AVs with different driving behaviors. After collecting the surrogate point clouds, the attacker derives the set of critical adversarial locations using our proposed algorithms. Finally, the attacker can launch the attacks by placing arbitrary objects around the derived critical adversarial locations. As long as these objects can reflect laser, the attack goal can be achieved. With the precalculated critical adversarial locations in a target driving environment, the attacker can quickly launch the attack when the victim AV is arriving. The derived locations does not need to be changed during the attack.

6 EXPERIMENTS IN DIGITAL WORLD

6.1 Experimental Setting

We first use one of the most common LiDAR detection models, i.e., PIXOR [63], as our target model. We train PIXOR with KITTI dataset [18] to detect the vehicles on the road. Since our attack goal here is to hide the target vehicle, we randomly select 200 frames (examples) from the dataset and consider one target vehicle in front of the victim in each frame. Then we take the selected point clouds as the surrogate data D^* and generate critical adversarial locations using our algorithms. Specifically, when finding the critical locations, we only consider an area with the size of $4m * 4m * 1m$ above the target vehicle. The reason why we select this area is that the inserted objects in this area will not be blocked by other objects and placing objects in this area is more feasible. When calculating the adversarial scores, the K is set to 4. The detection confidence threshold is set to 0.7. After finding the critical locations, we randomly generate a point cluster around each generated location to evaluate the attack performance. We repeat this random generation for 100 times and calculate the average result. The size of each random point cluster, which is defined as the maximum distance among all points, is set to $0.2m$. The number of points in each cluster is 4.

To evaluate the attack performance, we calculate the attack success rate, which is defined as the percentage of the examples (LiDAR frames) that are successfully attacked among all the examples. An example is successfully attacked when the target vehicle is successfully hidden. Please note that here we consider one target vehicle in front of the victim for each example. To demonstrate the attack feasibility and stealthiness, we also evaluate the number of critical adversarial locations that are required to achieve the attack goal. In addition, in this paper, the proposed adversarial location-based attack using the basic probing algorithm described in Section 5.1 is called **AdvLo**. The proposed attack using the efficient probing algorithm described in Section 5.3 is called **AdvLo-EP**.

6.2 Overall Performance

In this experiment, we evaluate the performance of our algorithms on the selected examples under different values of Q, i.e., the number

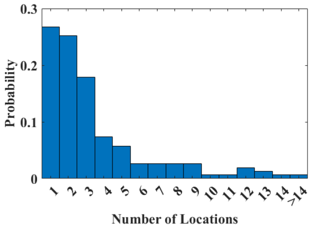


Figure 2: Histogram of number of critical adversarial locations.

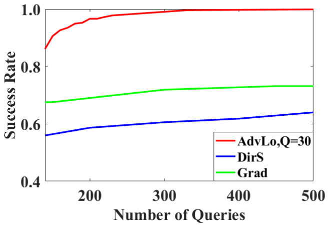


Figure 3: Attack efficiency of AdvLo.

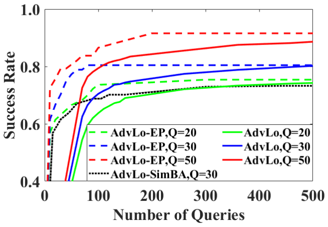


Figure 4: Attack efficiency of AdvLo-EP.

of the probed locations in each iteration in the Location Probing step. We set the number of probing iterations P to 1000 and vary Q from 20 to 50. In Table 1, we show the average attack success rate and the average number of critical adversarial locations generated by AdvLo and AdvLo-EP. The results show that AdvLo can achieve 93% success rate when Q is 50. In addition, we can observe that both AdvLo and AdvLo-EP can achieve similar success rates with similar numbers of critical locations under different values of Q. We can also find that more searching locations in Location Probing step results in higher attack success rate. This is mainly because we can probe more locations when Q is large, which increases the probability of finding the critical adversarial locations.

Table 1: The average attack success rate and number of critical adversarial locations

	Average Success Rate		Average Number of Locations	
Q	AdvLo	AdvLo-EP	AdvLo	AdvLo-EP
20	0.77	0.78	3.07	3.36
30	0.82	0.81	4.24	4.15
40	0.84	0.86	4.87	4.82
50	0.93	0.92	5.28	5.33

To show the feasibility of our attack, we further investigate the number of critical adversarial locations required to achieve the attack goal. Figure 2 shows the histogram of the number of critical locations in all successfully attacked examples based on AdvLo. We can observe that over a half of the examples require no more than 2 critical adversarial locations, and around 70% of the examples require no more than 3 locations. The results indicate that the attacker only needs to use a few arbitrary objects (one object for each location) to achieve the attack goal, which shows high feasibility of our proposed attack.

6.3 Attack Efficiency

Next, we demonstrate the efficiency of our proposed two-step framework. We first evaluate the efficiency of AdvLo and compare it with directly searching in one step. Since the computation time is determined by the number of queries, we measure the number of queries required to generate critical adversarial locations based on which the attack goal can be achieved. In this experiment, we select the examples that are successfully attacked with less than 5 adversarial locations based on the proposed algorithm, and set the value of Q as 30. Here we consider two baseline methods that directly search for the critical location set. The first baseline method, referred as DirS, directly searches for X containing 5 locations that minimizes the value of $L(X)$ in Eq. (2) through random searching algorithm. The second baseline method use gradient estimation and fast gradient descent to directly search for 5 locations X that minimizes $L(X)$. Figure 3 shows the attack success rate with respect to the number of queries. We can observe that compared with the baseline methods, our proposed two-step framework requires much less quires to successfully attack all the selected examples. The results also show that directly finding the critical locations is difficult due to large searching space. Our proposed two-step framework, that first probes a large number of locations and then selects the most critical ones, is more efficient and effective. We also calculate the required time for AdvLo to generate the adversarial locations. In our experiment, the computation time for each query is around $0.56s$. Here we use parallel computing in the Location Probing step with 10 processes on 1 GPU. When Q is 30, to achieve an 80% success rate, the average time of generating the adversarial locations for each example is $92s$, which is sustainable considering that the calculation is conducted in an offline manner.

We then evaluate the efficiency of the proposed efficient probing algorithm AdvLo-EP. Specifically, we compare the attack success rate of AdvLo-EP with that of AdvLo under different numbers of queries required in the Location Probing step. Here we vary the value of Q from 20 to 50. In addition, we implement a baseline that uses SimBA algorithm [21] in the Location Probing step, denoted as AdvLo-SimBA. SimBA is a query-efficient black-box algorithm, which repeatedly picks a random searching direction and adds the vector if it points towards the decision boundary, otherwise subtracts it. The baseline AdvLo-SimBA uses SimBA to search for the location set $\hat{X}$ in the Location Probing step. We report the results in Figure 4. We can observe that the proposed Efficient Probing algorithm AdvLo-EP can always achieve better attack success rate with less queries under different values of Q. The results also show that AdvLo-EP performs better than AdvLo-SimBA. This is because SimBA may get stuck in local minimums. We further calculate the required time for AdvLo-EP to generate the adversarial locations. Here parallel computing is not used. When Q is 30, to achieve an 80% success rate, the average computation time is $44.7s$. To sum up, our proposed attack framework using the efficient probing algorithm can derive the critical adversarial locations more efficiently compared with random sampling and other searching baselines.

6.4 Robustness Analysis

Effect of object size. In this paper, we aim to perform the proposed attack with arbitrary objects that can reflect laser. To demonstrate the robustness of our proposed attack to object size (or shape), we analyze the effect of object size on the attack success rate. For the 200 selected samples, we generate random point cluster with

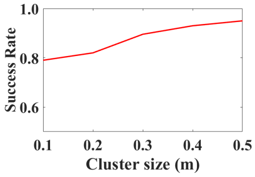


Figure 5: Effect of the object size.

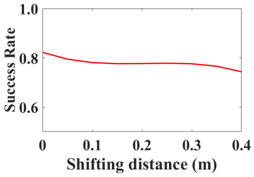


Figure 6: Effect of location errors.

different sizes from 0.1m to 0.5m, and insert the clusters to the critical adversarial locations derived by AdvLo where $Q = 30$. The number of points in each cluster is calculated as $(Size * 10)^2$. We then calculate attack success rate for different cluster sizes. The experiment is repeated for 100 times and we report the average result in Figure 5. We can observe that the attack success rate increases as the cluster size increases, and even reaches 94% when the cluster size is 0.5m. This is because larger point cluster contains more points and can cover more adversarial region, thus has more adversarial effect to the model's output. It is also intuitive that placing larger objects around the vehicle is more likely to distort the shape of the target vehicle and result in failed detection. The results in this experiment show that increasing object size does not degrade the performance of our attack, which means our attack is robust to the size of the adopted object.

Effect of location errors of objects. In physical world deployment, it is usually difficult to place objects precisely at the generated critical locations. Even after the deployment, the objects may also be displaced due to various factors such as the wind. In this experiment, we demonstrate the robustness of our attack to the location errors when placing the objects. Specifically, we analyze the variance of attack success rate when each random point cluster is shifted from the critical adversarial location with different distances. Here the critical adversarial locations are derived based on AdvLo when the value of Q is set as 30. Figure 6 shows the attack success rate when the shifting distance varies from 0.05m to 0.4m towards random directions. The size of each point cluster is 0.2m. We repeat the experiments for 100 times and calculate the average result. As shown in Figure 6, the attack success rate almost keeps the same when the shifting distance varies. This is because the region around the derived location is also adversarial. In other words, the adversarial location could be an adversarial region. This is also demonstrated in Section 8.2. So the attacker does not need to place the objects at the critical locations with high precision to achieve the attack goal.

6.5 Comparison with Existing Attack

To further demonstrate the advantage of our proposed attack, we compare it with the attack method proposed in [51] (denoted as **Phy-adv**), which has the same attack goal as ours, i.e., hiding the target vehicle by adding some adversarial objects. Phy-adv implements the attack by generating a 3D mesh and placing it on the rooftop of the target vehicle. Compared with Phy-adv, our proposed attack is not only more flexible and easier to perform, but also stealthier in practice.

In this experiment, we generate adversarial objects for the same 200 examples as previous experiments. For Phy-adv, we follow the setting described in [51]: the size of the object is set to $0.7 * 0.7 * 0.5m$, and the object is placed at the center of the rooftop of the target vehicle. To evaluate how easy and flexible our attack is, we compare the success rate of our attack with that of Phy-adv after shifting the adversarial objects and distorting their shape. Intuitively, if the attack method can tolerate larger shifting distances and more distortion of the shape, it will be easier to perform the attack and more flexible when choosing the adversarial objects. Because the attacker does not need to generate adversarial objects with special shape and size and put them on specified locations with high precision.

Table 2: Comparison with existing attack method.

	Shifting Distance		Shape Distortion	
Method	0.15m	0.2m	0.05m	0.1m
Phy-adv	0.43	0.40	0.54	0.48
AdvLo	**0.78**	**0.77**	**0.82**	**0.82**

We first consider two scenarios where the shifting distances of the adversarial objects are set to 0.15m and 0.2m, respectively. Then we distort the objects with 0.05m and 0.1m (without shifting the objects). Table 2 reports the attack success rates for different settings. Please note that when the shifting distance is 0 and there is no shape distortion, the success rate of our attack is 0.82 while that of Phy-adv is 0.77. The results show that the success rate of Phy-adv decreases greatly after shifting the adversarial objects. However, our proposed AdvLo still has good performance after shifting the objects. This indicates that our attack can tolerate larger location errors of the adversarial objects compared with Phy-adv. Similarly, when the shape of the object is distorted, which may be caused by manufacture error and imprecision of LiDAR scans, the attack success rate of Phy-adv decreases. However, our attack is not affected by the distortion. This is because the adversarial objects in our attack can be in arbitrary shape. The above results demonstrate that our attack is more flexible and easier to perform compared with Phy-adv.

In addition, our attack has better stealthiness because the attacker only needs to let the drones hover for a few seconds and fly away immediately after the attack. However, in Phy-adv, the attacker has to install the adversarial object on top of the vehicle, which can not be removed quickly. Placing such a specially shaped object on the car for a long time could be suspicious.

6.6 Performance on Different Detection Models

Besides PIXOR, we also evaluate the performance of the proposed attack on the following state-of-the-art LiDAR detection models:

VoxelNet [74]. VoxelNet is a voxel-based LiDAR object detection model that divides a point cloud into 3D voxels and learns the voxel's feature through a voxel feature encoding layer. A region proposal network is adopted to generate the detection results.

PointPillars [29]. PointPillars is also a voxel-based detection model. It divides the point cloud into vertical columns (pillars) and utilizes PointNets to learn their features. Standard 2D convolutional detection network can be used to generate bounding box proposals.

F-PointNet [34]. F-PointNet is a point-based detection model utilizing PointNet-like network to segment foreground points from background in a 3D bounding frustum, which is extracted from 2D image detection results. A box regression PointNet model is used to generate bounding box proposals from the foreground points.

Table 3: The average attack success rate on different models.

Models	Average Success Rate	Average Number of Locations
F-PointNet	0.89	2.99
VoxelNet	0.71	5.57
PointPillars	0.74	6.02

In this experiment, we set the values of Q and P as 50 and 5000, respectively. We then randomly select 100 examples from KITTI dataset and use our algorithm AdvLo to generate critical adversarial locations on each example. Table 3 reports the average attack success rate and average number of critical locations for each detection model. We can observe that the attack against F-PointNet has the best performance with 89% success rate and 3 critical locations required on average. The attack performance of the other two models are similar, and their attack success rates are more than 70%. These results show that our attack framework is general enough to be applied to attack different LiDAR perception models.

7 EXPERIMENTS IN PHYSICAL WORLD

7.1 Experimental Setting

In this section, we evaluate the attack effectiveness and feasibility in physical world. The attack goal is to hide the target vehicle from the victim LiDAR detection system. To achieve the goal, we first imitate the victim AV and collect the surrogate point cloud data using an Ouster OS1-64 LiDAR shown in Figure 7. The precision of the LiDAR is $1.5 - 5cm$. Its sensing range is $120m$, and its vertical field of view is 45 degree. In our experiment, we mount and level the LiDAR on top of a sedan vehicle using a tripod with suction cups. The height of LiDAR from ground surface is $1.8m$.

To improve the attack robustness under various driving conditions, we collect original data as input D^* by imitating the victim vehicle and moving forward in different directions in the given surrounding environment. The adversarial impact is measured by the summation of the $L(\cdot)$ values in Eq. (2) for all collected frames. Since the target vehicle (i.e., the vehicle in front of the victim vehicle) may move during the attack, we propose to find the critical adversarial locations in the coordinate of the target vehicle so that we can achieve the attack goal in all collected frames. In this way, we can generate relative locations to the target vehicle's center and achieve constant attack by inserting objects that can move with the target vehicle. In our experiment, we constrain the size of the searching area to $4m * 4m * 1m$ above the car, whose center is the center of the target vehicle. As shown in Figure 8b and 8f, we evaluate the attack in two real-world scenarios where the black car is the target vehicle. We use drones as the objects to be placed around the generated critical locations. As we discussed, drone is a good choice for launching the attack due to its stealthiness and flexibility. It is easy to control drones to fly to the given locations, and move with the target vehicle to perform constant attacks.

Here we consider PIXOR as the target detection model. Based on the collected original point cloud data, we use the proposed algorithm AdvLo to derive two critical locations that can be used to hide the target vehicle from the victim AV. Thus, we only need two drones to achieve the attack goal. The two drones we use are DJI Phantom 4 pro and DJI Mavic pro. Since Mavic pro is too small and might not be able to create stable reflected LiDAR points especially when it is far away from the LiDAR, we hang a cardboard below it as shown in Figure 8b and 8f.

Figure 7: The Ouster OS1-64 LiDAR.

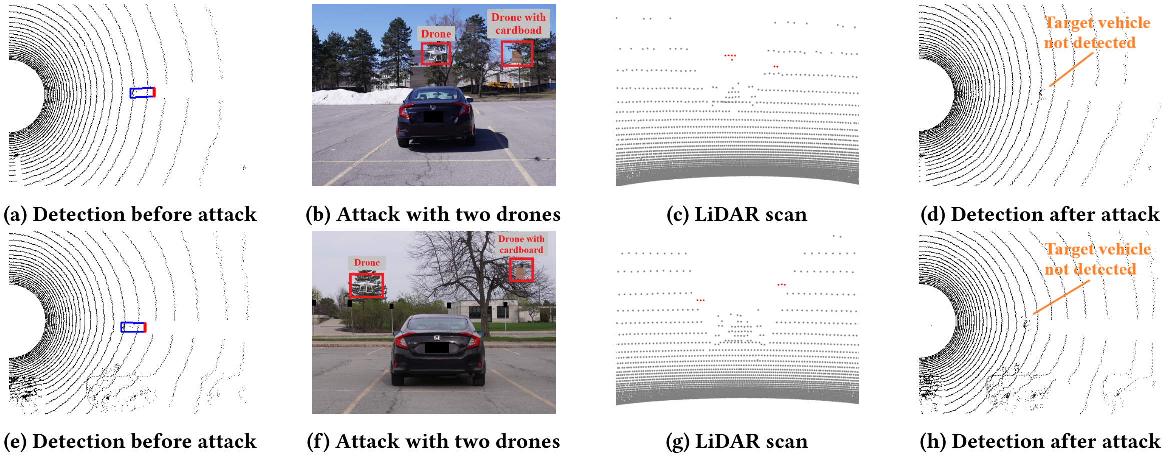
7.2 Results Visualization

Figure 8 shows the attack results in the two scenarios. Figure 8a and 8e show the detection results before the attack in bird-eye views, where the target vehicle is successfully detected and marked by a bounding box. The red side of the bounding box indicates the head of the detected vehicle. Figure 8b and 8f show the drones are hovering around the critical locations. In the point cloud scanned by the LiDAR, each drone creates a small point cluster with 3-5 points around the given locations, as shown by the red points in Figure 8c and 8g. Figure 8d and 8h show the detection results after the attack. We can observe that the target vehicle is not detected by the LiDAR perception system in both two scenarios, which demonstrate the effectiveness of our proposed attack in physical world.

7.3 Robustness in Physical World

Effect of driving directions and distances. To show the robustness of our attack, we also drive the victim AV in different directions towards the target vehicle and conduct continuous attack. In order to collect enough evaluation data, we drive the victim AV from $30m$ to $5m$ away from the target vehicle at the speed of around 2m/s in different directions (i.e., left side of the target vehicle, right side of the target vehicle, and directly behind the target vehicle). We conduct the attack in two different scenarios in Figure 8, and collect 801 LiDAR frames in total. The attack success rates for different directions and distances are shown in Table 4. For each distance range (e.g., $5 - 10m$), we report the average result over the frames collected in this range, and we can see that the attack success rates are similar when the victim drives in different directions. As the victim is driving close to the target, the attack success rate decreases. This is because the target vehicle generates more LiDAR points when it is closer to the LiDAR, which benefits the feature learning of the detection model. However, even when the victim vehicle is close to the target (e.g., $5 - 10m$), the success rate of our attack is still around 80% on average. In Figure 9, we visualize some examples of the attack results. Figure 9a, 9b, and 9c show the detection results when the victim AV approaches the target vehicle. Figure 9d, 9e, and 9f show the detection results when the victim AV drives in different directions. As we can see, the target vehicle is not detected in the shown 6 examples.

All the above results show that our proposed attack is not only effective and feasible, but also robust in real-world driving environment, which enables real-world continuous attack. The reason for the robustness of our attack is that we take into account different driving directions and distances when generating critical adversarial locations. As discussed in Section 4 and Section 5.4, the attacker



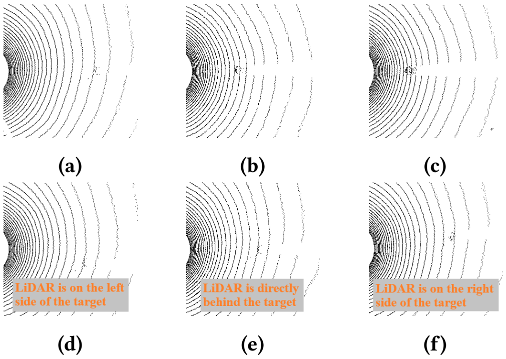
(a) Detection before attack **(b) Attack with two drones** **(c) LiDAR scan** **(d) Detection after attack**

(e) Detection before attack **(f) Attack with two drones** **(g) LiDAR scan** **(h) Detection after attack**

Figure 8: Attack using drones in two physical world scenarios.

Table 4: Success rates for different directions and distances.

	$5-10m$	$10-15m$	$15-20m$	$20-25m$	$25-30m$
Left	0.83	0.85	0.90	0.96	0.98
Behind	0.82	0.82	0.92	1.00	1.00
Right	0.79	0.85	0.93	1.00	0.97



(a) **(b)** **(c)**

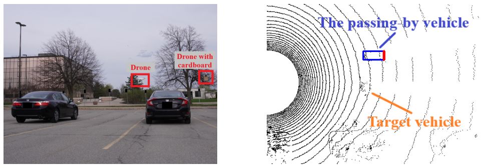
(d) **(e)** **(f)**

Figure 9: Detection results in different positions.

first imitates the possible behaviors of the victim AV and collect the surrogate point clouds D^* of the target driving environment. These point clouds are collected in different directions and from different distances. Then these point clouds are used to generate the critical adversarial locations. Thus, the derived locations are robust to different driving directions and distances.

Effect of passing-by vehicles. To further demonstrate the robustness of our proposed attack, we also study the effect of small environment changes on the attack performance. Specifically, we consider the most common scenario where there are other vehicles passing by. The passing-by vehicles would change the LiDAR point cloud and might influence the attack performance. We evaluate this effect by considering the same scenario in Figure 8f. To simulate a passing-by vehicle, we drive another car beside the target car as shown in Figure 10a. The critical adversarial locations are the same locations in Figure 8f.

Figure 10b shows the detection result after the attack. The target vehicle is still not detected, which means the passing-by vehicle has almost no effect on the performance. This is mainly because the edge of the target vehicle plays a critical role in the detection process, as discussed in Section 8.2, and the small environment change caused by a passing-by vehicle does not affect the geometric features around the target's edge. Additionally, in our real-world experiments, the critical adversarial locations are generated from large amount of original point cloud data collected under various driving conditions, which is robust to the small environment changes. Here the passing-by vehicle can be detected because we do not target on this vehicle. The results show that, by considering various driving conditions and collecting corresponding original point clouds, the attacker is able to perform robust attack against victim AVs in physical world.



(a) Passing-by car **(b) Detection result**

Figure 10: Detection result when another car passes by.

8 ANALYSIS FOR ADVERSARIAL LOCATIONS

8.1 Adversarial Region Map

We study the characteristics of adversarial locations by visualizing them using a heat map. Understanding the characteristics of the adversarial locations, such as their distribution in the 3D space, is essential for analyzing the vulnerability of the LiDAR object detection models and the possible security threats caused by adversarial attacks. It can also help us understand why and how these adversarial locations affect the model's output and which region around the target vehicle is more likely to be adversarial. In addition, the characteristics of the adversarial locations may also provide potential guidance for developing robust object detection models against adversarial attacks.

In Section 5, we discuss how to generate the critical adversarial locations and their adversarial scores. It is intuitive to generate the map using the adversarial scores. Although it is impractical

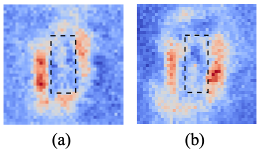


Figure 11: Average adversarial region map: (a) PIXOR, (b) VoxelNet.

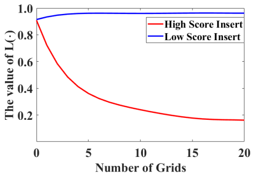


Figure 12: Inserting points based on adversarial region map.

to calculate the adversarial score of every location in the given region, we can sample a large number of locations and calculate their adversarial scores. Specifically, in each of the P iterations, we select Q random locations X^p from the given area and calculate $L(X^p)$. Based on Algorithm 1, we can calculate the adversarial scores of all the Q locations in each iteration p. We combine all the $Q * P$ locations together as well as their adversarial scores. Then we divide the space into small grids and the adversarial score of each grid is the average score of the selected locations in the grid.

8.2 Experiments for Adversarial Region Map

Next, we investigate the characteristics of the adversarial locations by visualizing the adversarial region maps. In this experiment, we consider the area around each target vehicle. Specifically, we generate the adversarial region maps within an $8m * 8m$ square area in bird's-eye views, whose center is the center of the target vehicle. The area is divided into $0.2m * 0.2m$ grids and each grid is assigned a value based on its adversarial score, as discussed in Section 8.1. Here we randomly select 100 examples (target vehicles) from KITTI dataset and take them as test data.

Visualization. In this experiment, we take PIXOR and VoxelNet as the target detection models. For each model, we first generate the adversarial region map for each of the 100 examples based on the method described in Section 8.1. Then we derive an average map by taking the average over all the generated adversarial region maps. Figure 11 shows the derived average maps for the two detection models. In this figure, the regions with high adversarial scores are highlighted using red color, and the deeper the color, the higher the adversarial score. Blue color indicates low adversarial score. The average location of the target vehicles is highlighted by a dashed bounding box. We can see that although the distributions of adversarial locations for the two detection models are different, most of the regions with high adversarial scores are around the edge of the target vehicles. This is mainly because the locations around a target vehicle's edge play a critical role in determining the shape/outline of the vehicle and their states can significantly affect the learned geometric features of the vehicle. We can also find that the center of the vehicle's top is less important than the edge. This is probably because the top of the vehicle is barely scanned by LiDAR in real-world driving scenarios and thus has fewer or even no LiDAR points, as shown by Figure 13. Thus, the edge regions can be used by the attacker to perform adversarial attacks against LiDAR perception models. Through changing the states of the locations (inserting points around them) in these regions, the attacker can "distort" the shape/outline of the target vehicle and change the learned geometric features of the target, which further misleads the detection.

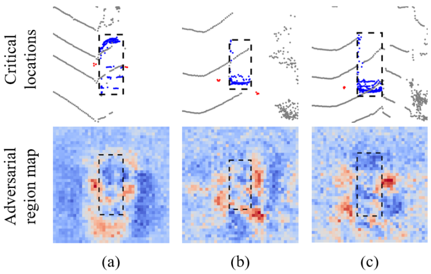


Figure 13: Examples of adversarial region maps.

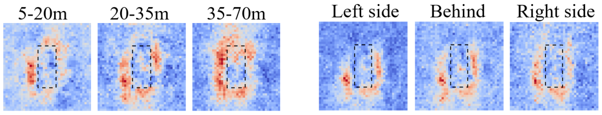
Figure 13 shows the adversarial region maps for three examples that are successfully attacked (each column describes one example). Here the detection model is PIXOR. In the first row of this figure, we show the bird's-eye view of the point cloud and the inserted random point clusters around critical adversarial locations. We also highlight the points belonging to the target vehicle with blue color and the inserted point clusters around critical locations with red color. The adversarial region map of each example is shown in the second row. The locations of the target vehicles are highlighted in both the region maps and the point clouds by dashed bounding boxes. The results show that the critical adversarial locations are always around the grids with high adversarial scores in the map.

In addition, the results in Figure 13 show that the critical locations are only in a few red grids. Take the map in the first column as an example, many grids are marked red but there is only one critical adversarial location in one of the grids. The reason why we can use a few critical adversarial locations to achieve the attack goal can be explained from two aspects.

On one hand, for deep neural networks, small changes on the input can be propagated to many feature layers and cause large changes on the final learned features [52]. For example, recent studies have found that changing one pixel in an image can significantly change the learned features and further change the classification results [43]. In LiDAR detection models, the neural network is usually designed to be very deep in order to capture both small-scale (local) features and large-scale (global) features. Thus, small changes on the input (i.e., inserting a few points at a few locations) may significantly change the final learned geometric features. To demonstrate the effect of the inserted points on the learned features, we visualize the difference between the feature maps before and after the attack, which is shown in Section B of the Appendix.

On the other hand, the LiDAR points of the target vehicles are usually sparse because of occlusion and missing reflection of laser signals, as shown in Figure 13. Due to the sparsity of the target's LiDAR points, inserting points around a few locations can be enough to distort the shape or outline of the target vehicle and further affect the geometric features. The experimental results in Table 4 and Figure 14a also show that it is easier for the attacker to perform the attack when the target vehicle is further from LiDAR, because the target generates less LiDAR points when the distance between the target and LiDAR is larger.

In Figure 13, we can also observe that the regions around the derived adversarial locations are also adversarial, which means nearby locations tend to share the same vulnerability as the derived adversarial locations. So the inserted points can tolerate shape



(a) Different distances **(b) Different directions**

Figure 14: Adversarial region maps under different settings.

distortions and location errors. This also explains why our attack can use arbitrary objects and is robust to location errors as long as they are around the derived adversarial locations.

Inserting points according to the map. To further validate the generated adversarial region map, we insert random points into the grids with high adversarial scores derived for PIXOR, and then calculate $L(\cdot)$ defined in Eq. (2). Specifically, we select a particular number of grids with the highest adversarial scores and randomly insert 4 points into each of these grids (each grid is 0.2m*0.2m). This operation is referred to as *high score insert*. We also randomly insert points into the grids with lowest adversarial scores in the same way, which is referred to as *low score insert*. Figure 12 shows the changes of the value of $L(\cdot)$ with respect to the number of grids in these two kinds of insertions. As expected, randomly inserting points into the adversarial regions with high adversarial scores results in the drop of confidence score (i.e., the value of $L(\cdot)$). Inserting points into the regions with low adversarial scores makes the confidence score increase slightly, which means the detection results of the target vehicles are improved. These results verify that the regions with high adversarial scores have negative effect on the detection results and the regions with low adversarial scores have positive effect on the detection results.

Adversarial locations under different settings. Next, we study the factors that may affect the adversarial locations. Here we still take PIXOR as an example.

We first study the effect of LiDAR's distance to the target vehicle. Specifically, we divide the examples into three groups based on the distance between the target vehicle and the LiDAR. Specifically, the distance ranges for the three groups are $5-20m$, $20-35m$, and $35-70m$, respectively. Then we generate the average adversarial region map for each group, which is shown in Figure 14a. We can see that although the distributions of adversarial locations in the three maps are different, there are some locations that have high adversarial scores in all the three maps. This indicates that we could find some adversarial locations that are critical even when the distance varies, which can help the attacker achieve continuous attack (as shown in Section 7.3). In addition, the results show that when the target is further from LiDAR, the adversarial region becomes larger, which is also demonstrated in Table 4.

We then study the effect of LiDAR's direction to the target vehicle. We divide the examples into three groups based on LiDAR's direction to the target, i.e., left side of the target, right side of the target, and directly behind the target. To better categorize these groups, here we consider the vehicles that are driving forward in front of the LiDAR. Figure 14b shows the average adversarial region maps for the three groups. We can see that the directions have a small effect on the distribution of adversarial locations.

In addition, we study the effect of other factors including the target vehicle's size and the street where the target vehicle is located. The experimental results are shown in Section C of the Appendix.

9 DISCUSSION

9.1 Potential Defense Strategies

Defense based on adversarial location. Our investigation on the adversarial region map shows that the locations around the edge of the target vehicle play critical roles in determining the outputs of LiDAR perception models. The attacker can take them as adversarial locations and use them to perform adversarial attacks by simply changing their states. To defend against such attacks, a potential strategy is to prevent the attacker from deriving these adversarial locations. In our attack method, the adversarial locations are generated by finding the locations with high adversarial scores. Thus, we propose to reduce the adversarial scores of the locations around the edge of the target vehicle when training the detection model so that the attacker cannot identify these locations in the follow-up attacks. According to Eq. (3) and Eq. (5), the adversarial score of the locations in X is determined by function $L(X)$. The larger the value of $L(X)$, the less the adversarial scores of the locations in X. Our goal in this defense strategy is to maximize the value of $L(X)$ given that X is the locations around the edge of the target vehicle. Specifically, we integrate $L(X)$ into the loss function adopted by the original LiDAR detection model. However, since $L(X)$ is not differentiable, we approximate it using a differentiable function: $L'(X) = \max_{y \in Y} \max(1-(R(y, y^*)-R_t), 0) * C(y)$, where $R(y, y^*)$ measures the distance between the output bounding box proposal y and the ground truth bounding box y^*. R_t is the distance threshold to check whether the bounding box proposal y is relevant to the target vehicle y^*. In addition, we consider all the vehicles in the training data and maximize the $L(X)$ for each vehicle. Then we train the LiDAR perception model using the following loss function.

$$\hat{L} = L_{pred} - \beta \sum_{y^* \in Y^*} L', \tag{7}$$

where Y^* contains the ground truth bounding boxes of all the vehicles in the training data, and L_{pred} is the original loss function adopted by the detection model. β is a trade-off parameter.

To evaluate the proposed defense strategy, we choose PIXOR as the detection model. We train PIXOR using the proposed loss function $\hat{L}$, and β is set to 0.01. For each vehicle, X is the location set containing 10 locations which are randomly chosen around the edge of the vehicle. Then we generate the adversarial locations for the same 200 examples that are used in Section 6 to attack the new detection model. The attack success rate is reduced to 0.34, which shows that the proposed defense strategy is effective and it can make the detection model more robust to mitigate the adversarial attacks. However, we find that by increasing the number of probed locations Q in the attack, the attacker can still achieve a good attack success rate. When Q is increased to 60, the attack success rate is increased to 0.59. Besides, we find that increasing the number of locations in X may degrade the detection accuracy when there is no attack. Thus, there is a trade off between the defense performance and detection performance.

Other defense strategies. Another potential defense strategy against the proposed attack is to train the model with adversarial examples (i.e., adversarial training). To evaluate the performance of this defense strategy, we use PIXOR as the target detection model and retrain the model by inserting adversarial point clusters into the training data. Specifically, when training the new model, we randomly select one target vehicle in each LiDAR frame, and insert 10 random point clusters around the target vehicle into the original point cloud. Our experimental results show that the attack success rate on the retrained model can be reduced to 75%, which means the adversarial training can degrade the performance of our proposed attack. However, our proposed attack can still achieve high success rate even after adversarial training. Besides, we can use LiDAR together with other types of sensors (e.g., camera and radar) and detect objects based on sensor fusion. However, such defense strategy requires additional sensors and increases the cost of autonomous vehicle systems. In addition, recent studies have found that image perception systems and radar are also vulnerable to some attacks [27, 64]. It is entirely possible to attack multiple sensors simultaneously and fool the sensor fusion systems [6, 50].

9.2 Limitations and Future Work

One limitation of our attack is that the derived critical adversarial locations on one street may not be useful on another street. Thus, the attacker needs to recalculate the critical adversarial locations if he wants to perform the attack on a new street. To address this problem, in Section D of the Appendix, we provide a preliminary study on the universal attack, where the derived adversarial locations can be used to fool the detection models on any streets. Further study on the universal attack will be our future work.

Another limitation of our method is that in some cases the attacker needs to place adversarial objects at more than three critical adversarial locations to achieve the attack goal. If the attacker uses drones to perform the attack, how to simultaneously control so many drones is a practical challenge. However, drone localization and communication techniques have been well developed, and multiple drones are able to cooperate with each other to perform some complicated tasks such as drone display [48, 61]. In our future work, we will combine such techniques with the proposed attack.

10 RELATED WORK

10.1 Vehicular System Security

The security issues of vehicular systems have drawn significant attention [11–13, 25, 26, 39, 40, 47, 54, 55, 71, 73], and there are extensive prior works studying the security vulnerabilities of AVs [14, 16, 17, 22, 24, 37, 41, 44, 46, 56, 60]. Although different methods have been developed to attack the perception systems of AVs [38, 62], most of them focus on camera-based perception [27, 59, 64]. There are only a few works that study the security threats to LiDAR-based perception systems.

Currently, there are mainly two categories of attacks against LiDAR perception systems: laser-based attack and object-based attack. For the laser-based attack methods [7, 23, 45], the attacker achieves the attack goal through strategically transmitting laser signals to the victim vehicle's LiDAR sensor. Although this kind of attacks can achieve good performance, they require the attacker to aim at the LiDAR with high precision, which is usually difficult to achieve when the vehicle is moving in physical world. In addition, the laser signals need to be generated by some special devices, which limits the flexibility of this kind of attacks. For the object-based attack methods, the attacker achieves the attack goal by generating some adversarial objects and placing them on the rooftop of the target vehicle [3, 50, 51] or on the road [8]. However, these objects usually have specific shapes and sizes, so their attack flexibility is limited. The attacker can only use the specifically shaped object to attack the victim AV on one specific detection model, one specific vehicle type, or one specific target scenario. In addition, these adversarial objects are in abnormal shapes which can be suspicious to human eyes. And the inaccuracies when crafting these objects in physical world as well as the noises of LiDAR scanning may fail the attacks.

Different from the above works, our goal in this paper is to design an easier and more flexible way and use arbitrary objects to attack LiDAR object detection in autonomous driving. Our attack can be implemented with any objects that have good stealthiness (e.g., drones, traffic signs, and advertisement board), as long as these objects can reflect laser. More importantly, it is easy to perform our proposed attacks in physical world. The attacker only needs to place some objects around the derived adversarial locations to achieve the attack goal.

10.2 3D Adversarial Attacks

3D adversarial attacks have been widely studied to to fool point cloud classification models in digital world [30, 57, 58, 67, 69, 70, 72]. However, point cloud classification is different from LiDAR object detection. The goal of point cloud classification is to derive a label (e.g., chair and desk) for the whole point cloud while LiDAR object detection aims to find the locations/orientations of the targets such as vehicles and generate a bounding box for each vehicle. Thus, existing 3D adversarial attack techniques cannot be directly applied to our problem. In addition, the adversarial examples generated by these techniques are not always physically realizable because they mainly focus on digital world. In contrast, in this paper, we study how to perform practical and effective adversarial attacks in real-world driving environments.

11 CONCLUSIONS

In this paper, we investigate the possibility of using arbitrary objects with reflective surface to attack LiDAR perception systems. Specifically, we propose a novel attack framework based on which the attacker can identify the least number of critical adversarial locations in the physical space. By placing some objects around these locations, the attacker can easily fool the LiDAR perception system. We evaluate the proposed attack framework on both a public LiDAR point cloud dataset and a real-world LiDAR perception testbed. The experimental results show that our proposed attack can achieve 93% success rate and has strong robustness. We also successfully attack the real-world LiDAR perception system using only two commercial drones based on the proposed attack framework. In addition, we visualize the distribution of adversarial locations and discuss their characteristics.

12 ACKNOWLEDGMENTS

We thank our anonymous reviewers for their insightful comments and suggestions on this paper. This work was supported in part by the US National Science Foundation under grant CNS-1626374, CNS-2120369, CNS-1737590, CNS-1652503, and ECCS-2028872.

REFERENCES

[1] [n. d.]. Baidu Apollo. https://apollo.auto/.

[2] [n. d.]. Driverless taxis to be available in Phoenix 'in weeks'. https://www.bbc.com/news/technology-54476524.

[3] Mazen Abdelfattah, Kaiwen Yuan, Z Jane Wang, and Rabab Ward. 2021. Towards Universal Physical Attacks On Cascaded Camera-Lidar 3D Object Detection Models. *arXiv preprint arXiv:2101.10747* (2021).

[4] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. Poster: Recovering the Input of Neural Networks via Single Shot Side-channel Attacks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2657–2659.

[5] Sebastian P Bayerl, Tommaso Frassetto, Patrick Jauernig, Korbinian Riedhammer, Ahmad-Reza Sadeghi, Thomas Schneider, Emmanuel Stapf, and Christian Weinert. 2020. Offline model guard: Secure and private ML on mobile devices. In *Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 460–465.

[6] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks. *arXiv preprint arXiv:2106.09249* (2021).

[7] 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.

[8] 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).

[9] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of 2017 IEEE symposium on security and privacy (sp)*. IEEE, 39–57.

[10] Kuei-Huan Chang, Po-Hao Huang, Honggang Yu, Yier Jin, and Ting-Chi Wang. 2020. Audio Adversarial Examples Generation with Recurrent Neural Networks. In *Proceedings of the 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 488–493.

[11] Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Z Morley Mao, and Henry X Liu. 2018. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Proceedings of the Network and Distributed System Security Symposium*.

[12] Anupam Das, Martin Degeling, Xiaoyou Wang, Junjue Wang, Norman Sadeh, and Mahadev Satyanarayanan. 2017. Assisting users in a world full of cameras: A privacy-aware infrastructure for computer vision applications. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1387–1396.

[13] Lucas Davi, Denis Hatebur, Maritta Heisel, and Roman Wirtz. 2019. Combining safety and security in autonomous cars using blockchain technologies. In *Proceedings of the International Conference on Computer Safety, Reliability, and Security*. Springer, 223–234.

[14] Bruce DeBruhl and Patrick Tague. 2018. Optimizing a MisInformation and MisBehavior (MIB) Attack Targeting Connected Cars. en. In *Proceedings of the IEEE Connected and Automated Vehicles Symposium (CAVS)*, Vol. 5.

[15] Ghada Dessouky, Patrick Jauernig, Nele Mentens, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2020. AI Utopia or Dystopia-On Securing AI Platforms. In *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[16] Kevin Fu and Wenyuan Xu. 2018. Risks of trusting the physics of sensors. *Commun. ACM* 61, 2 (2018), 20–23.

[17] Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, Chen, and Qi Alfred. 2020. A comprehensive study of autonomous vehicle bugs. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 385–396.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[19] Jairo Giraldo, Alvaro Cardenas, Murat Kantarcioglu, and Jonathan Katz. 2020. Adversarial Classification Under Differential Privacy. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium 2020*.

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

[21] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. 2019. Simple black-box adversarial attacks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2484–2493.

[22] Jun Han, Madhumitha Harishankar, Xiao Wang, Albert Jin Chung, and Patrick Tague. 2017. Convoy: Physical context verification for vehicle platoon admission. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*. 73–78.

[23] Zhongyuan Hau, Kenneth T Co, Soteris Demetriou, and Emil C Lupu. 2021. Object removal attacks on lidar-based 3d object detectors. *arXiv preprint arXiv:2102.03722* (2021).

[24] David Ke Hong, John Kloosterman, Yuqi Jin, Yulong Cao, Qi Alfred Chen, Scott Mahlke, and Z Morley Mao. 2020. AVGuardian: Detecting and Mitigating Publish-Subscribe Overprivilege for Autonomous Vehicle Systems. In *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 445–459.

[25] Shengtuo Hu, Qi Alfred Chen, Jiwon Joung, Can Carlak, Yiheng Feng, Z Morley Mao, and Henry X Liu. 2020. CVShield: Guarding Sensor Data in Connected Vehicle with Trusted Execution Environment. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*. 1–4.

[26] Kai Jansen, Matthias Schäfer, Daniel Moser, Vincent Lenders, Christina Pöpper, and Jens Schmitt. 2018. Crowd-gps-sec: Leveraging crowdsourcing to detect and localize gps spoofing attacks. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1018–1031.

[27] Yunhan Jia Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei Wei. 2020. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

[28] Kaidi Jin, Tianwei Zhang, Chao Shen, Yufei Chen, Ming Fan, Chenhao Lin, and Ting Liu. 2020. A unified framework for analyzing and detecting malicious examples of dnn models. *arXiv preprint arXiv:2006.14871* (2020).

[29] 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.

[30] Daniel Liu, Ronald Yu, and Hao Su. 2019. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *Proceedings of 2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2279–2283.

[31] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, Thomas Schneider, et al. 2021. FLGUARD: Secure and Private Federated Learning. *arXiv preprint arXiv:2101.02281* (2021).

[32] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2016. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697* 1, 2 (2016), 3.

[33] 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 2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[34] 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.

[35] 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.

[36] 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 *Proceedings of the Advances in neural information processing systems*. 5099–5108.

[37] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. 2020. {SAVIOR}: Securing Autonomous Vehicles with Robust Physical Invariants. In *Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20)*. 895–912.

[38] Kui Ren, Qian Wang, Cong Wang, Zhan Qin, and Xiaodong Lin. 2019. The security of autonomous driving: Threats, defenses, and future directions. *Proc. IEEE* 108, 2 (2019), 357–372.

[39] Neetesh Saxena and Bong Jun Choi. 2016. Authentication scheme for flexible charging and discharging of mobile vehicles in the V2G networks. *IEEE Transactions on Information Forensics and Security* 11, 7 (2016), 1438–1452.

[40] Neetesh Saxena, Santiago Grijalva, Victor Chukwuka, and Athanasios V Vasilakos. 2017. Network security and privacy challenges in smart vehicle-to-grid. *IEEE Wireless Communications* 24, 4 (2017), 88–98.

[41] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. 2020. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under {GPS} Spoofing. In *Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20)*. 931–948.

[42] 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.

[43] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[44] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2020. Light commands: laser-based audio injection attacks on voice-controllable systems. In *Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20)*. 2631–2648.

[45] 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 *Proceedings of 29th {USENIX} Security Symposium ({USENIX} Security 20)*. 877–894.

[46] Mingshun Sun, Ali Al-Hashimi, Ming Li, and Ryan Gerdes. 2020. Impacts of constrained sensing and communication based attacks on vehicular platoons. *IEEE transactions on vehicular technology* 69, 5 (2020), 4773–4787.

[47] Mingshun Sun, Ming Li, and Ryan Gerdes. 2017. A data trust framework for VANETs enabling false data detection and secure vehicle tracking. In *Proceedings of the 2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.

[48] Anam Tahir, Jari Böling, Mohammad-Hashem Haghbayan, Hannu T Toivonen, and Juha Plosila. 2019. Swarms of unmanned aerial vehicles—a survey. *Journal of Industrial Information Integration* 16 (2019), 100106.

[49] 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.

[50] James Tu, Huichen Li, Xinchen Yan, Mengye Ren, Yun Chen, Ming Liang, Eilyan Bitar, Ersin Yumer, and Raquel Urtasun. 2021. Exploring Adversarial Robustness of Multi-Sensor Perception Systems in Self Driving. *arXiv preprint arXiv:2101.06784* (2021).

[51] 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.

[52] Danilo Vasconcellos Vargas and Jiawei Su. 2020. Understanding the one-pixel attack: Propagation maps and locality analysis. In *Proceedings of the CEUR Workshop Proceedings*, Vol. 2640. CEUR-WS.

[53] Qian Wang, Baolin Zheng, Qi Li, Chao Shen, and Zhongjie Ba. 2020. Towards Query-Efficient Adversarial Attacks Against Automatic Speech Recognition Systems. *IEEE Transactions on Information Forensics and Security* 16 (2020), 896–908.

[54] Shu Wang, Jiahao Cao, Xu He, Kun Sun, and Qi Li. 2020. When the Differences in Frequency Domain are Compensated: Understanding and Defeating Modulated Replay Attacks on Automatic Speech Recognition. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1103–1119.

[55] Shu Wang, Jiahao Cao, Kun Sun, and Qi Li. 2020. {SIEVE}: Secure In-Vehicle Automatic Speech Recognition Systems. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*. 365–379.

[56] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. 2020. Plug-N-pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT. In *Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20)*. 949–965.

[57] Matthew Wicker and Marta Kwiatkowska. 2019. Robustness of 3d deep learning in an adversarial setting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11767–11775.

[58] 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.

[59] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. 2019. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6898–6907.

[60] Wenyuan Xu, Chen Yan, Weibin Jia, Xiaoyu Ji, and Jianhao Liu. 2018. Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles. *IEEE Internet of Things Journal* 5, 6 (2018), 5015–5029.

[61] Wataru Yamada, Kazuhiro Yamada, Hiroyuki Manabe, and Daizo Ikeda. 2017. iSphere: self-luminous spherical drone display. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 635–643.

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

[63] 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.

[64] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Tsung-Yi Ho, and Yier Jin. 2020. Beyond Digital Domain: Fooling Deep Learning Based Recognition System in Physical World. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1088–1095.

[65] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* 30, 9 (2019), 2805–2824.

[66] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 103–117.

[67] Qiang Zhang, Jiancheng Yang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. 2019. Adversarial attack and defense on point sets. *arXiv preprint arXiv:1902.10899* (2019).

[68] Zihan Zhang, Mingxuan Liu, Chao Zhang, Yiming Zhang, Zhou Li, Qi Li, Haixin Duan, and Donghong Sun. [n. d.]. Argot: Generating Adversarial Readable Chinese Texts. ([n. d.]).

[69] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. 2020. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1201–1210.

[70] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. 2019. Pointcloud saliency maps. In *Proceedings of the IEEE International Conference on Computer Vision*. 1598–1606.

[71] Jinli Zhong, Suguo Du, Lu Zhou, Haojin Zhu, Fan Cheng, Cailian Chen, and Qingshui Xue. 2017. Security modeling and analysis on intra vehicular network. In *Proceedings of the 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE, 1–5.

[72] Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and Nenghai Yu. 2020. LG-GAN: Label Guided Adversarial Network for Flexible Targeted Attack of Point Cloud Based Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10356–10365.

[73] Lu Zhou, Le Yu, Suguo Du, Haojin Zhu, and Cailian Chen. 2018. Achieving differentially private location privacy in edge-assistant connected vehicles. *IEEE Internet of Things Journal* 6, 3 (2018), 4472–4481.

[74] Yin Zhou and Oncel Tuzel. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4490–4499.

APPENDIX

A EFFICIENT PROBING ALGORITHM

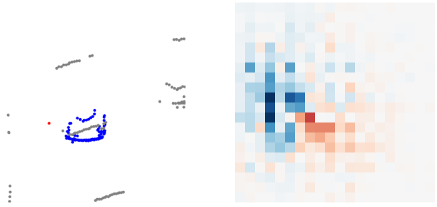
The algorithm of the Efficient Probing is summarized in Algorithm 3.

Algorithm 3: Efficient Probing algorithm

```
Input: the surrogate point cloud $D^*$; parameter $P$, $Q$, and $S$.
Initial $X^{all} \leftarrow \emptyset$, $X^{previous} \leftarrow \emptyset$;
Initial $X' \leftarrow$ random $Q$ locations from uniform distribution;
// Probing based on adversarial scores
for each iteration $p \in \{1, 2, ..., P\}$ do
    repeat
        Randomly generate $Q$ locations $\hat{X}$ based on Gaussian
          distribution, the means of the Gaussian distribution are
          the xyz-coordinates of $X'$;
    until $L(\hat{X}) < L(X^{previous})$;
    calculate approximated adversarial scores of each location in $\hat{X}$:
      $\tilde{W}(x), x \in \hat{X}$;
    $X^{all} \leftarrow X^{all} + \hat{X}$;
    update adversarial scores of all the locations in $X^{all}$;
    $X' \leftarrow$ the top $S$ locations with the highest adversarial scores
      among all previously probed locations $W(X^{all})$;
    $X^{previous} \leftarrow \hat{X}$;
end
Output: location set $\hat{X}$
```

B EFFECT OF THE INSERTED POINTS ON THE LEARNED FEATURES

To demonstrate the effect of the inserted points on the learned features, we visualize the difference between the feature maps before and after the attack. Figure 15a shows the target vehicle (the blue points) and the derived adversarial location (the red point), and Figure 15b shows the feature map difference around the target vehicle, which is derived by subtracting the feature map without inserting any point from the map after inserting one point. Each pixel in Figure 15b indicates the change of feature values around the corresponding locations in Figure 15a. The red color represents negative value and the blue color represents positive value. As we can see, inserting one point around the derived adversarial location can significantly affect the features around the target.



(a) (b)

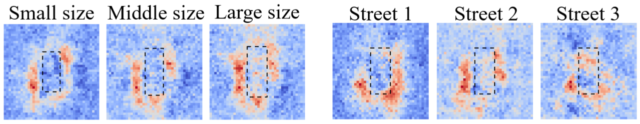
Figure 15: The changes of the feature map after inserting one point around the derived adversarial location.

C ADVERSARIAL LOCATIONS UNDER DIFFERENT SETTINGS

Here we study another two factors that may affect the adversarial locations. The two factors are the target vehicle's size and the street where the target vehicle is located.

To study the effect of the target vehicle's size, we divide the examples into three groups (i.e., small size, middle size, and large size). The size of a vehicle is defined by the product of its width, length, and height. As shown in Figure 16a, the distribution of adversarial locations for different sizes of target vehicle are similar, and most of the locations are close to the edge of the vehicles even when the size of the target vehicle varies.

For the streets where the target vehicle is located, we select three streets from the examples and generate the maps for the selected targets on each street. For each street, we select target vehicles that have similar sizes and distances to LiDAR. As shown in Figure 16b, the distributions of adversarial locations for the three streets are different. This is mainly because the detection model takes the LiDAR frames as input, and the LiDAR frames from different streets could be quite different, which affects the learned geometric features.



(a) Different sizes (b) Different streets

Figure 16: Adversarial region maps under different settings.

D UNIVERSAL ATTACK

In this paper, we also study the universal attack, where the attacker derives some *universal critical locations* based on which he can achieve the attack goal for any target vehicle in any scenario. These universal critical locations are the relative locations to the target vehicle's center. To generate universal critical locations, we use the whole dataset as the original data and measure the adversarial impact of locations using the summation of the values of $L(\cdot)$ for all the examples.

Based on the findings in Section 8.2, we consider the area around the edge of the vehicle, because it is more critical to the detection. Specifically, we search for the critical locations in a $2L * 2W * 1m$ area above the vehicle, where L and W are the average length and width of the vehicles. In the Location Probing step, the probability of generating probed locations around the edge of the vehicle is assigned higher than other areas. We set Q to 50, and P to 5000.

We conduct experiment on the KITTI dataset and the result show that the attack success rate of the universal attack is 72.69% when the number of critical locations is 7. This indicates the attacker can use 7 objects to perform the attack and may cause potential collisions in any scenario. Please note that this is the maximum number of locations the attacker needs to achieve the attack goal in all scenarios. In some scenarios, if the attacker knows the surrounding environment before launching the attack, he could use only some of these universal critical locations. Specifically, he could obtain the surrogate LiDAR point cloud data by following the same process in Section 7, and then select some universal critical locations using the Location Selection algorithm in Section 5.2.