Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving

Yulong Cao
University of Michigan
yulongc@umich.edu

Chaowei Xiao
University of Michigan
xiaocw@umich.edu

Benjamin Cyr
University of Michigan
bencyr@umich.edu

Yimeng Zhou
University of Michigan
yimzhou@umich.edu

Won Park
University of Michigan
wonpark@umich.edu

Sara Rampazzi
University of Michigan
srampazz@umich.edu

Qi Alfred Chen
University of California, Irvine
alfchen@uci.edu

Kevin Fu
University of Michigan
kevinfu@umich.edu

Z. Morley Mao
University of Michigan
zmao@umich.edu

ABSTRACT

In Autonomous Vehicles (AVs), one fundamental pillar is *perception*, which leverages sensors like cameras and LiDARs (Light Detection and Ranging) to understand the driving environment. Due to its direct impact on road safety, multiple prior efforts have been made to study its the security of perception systems. In contrast to prior work that concentrates on camera-based perception, in this work we perform the first security study of LiDAR-based perception in AV settings, which is highly important but unexplored. We consider LiDAR spoofing attacks as the threat model and set the attack goal as spoofing obstacles close to the front of a victim AV. We find that blindly applying LiDAR spoofing is insufficient to achieve this goal due to the machine learning-based object detection process. Thus, we then explore the possibility of strategically controlling the spoofed attack to fool the machine learning model. We formulate this task as an optimization problem and design modeling methods for the input perturbation function and the objective function. We also identify the inherent limitations of directly solving the problem using optimization and design an algorithm that combines optimization and global sampling, which improves the attack success rates to around 75%. As a case study to understand the attack impact at the AV driving decision level, we construct and evaluate two attack scenarios that may damage road safety and mobility. We also discuss defense directions at the AV system, sensor, and machine learning model levels.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; • **Computer systems organization → Neural networks**.

KEYWORDS

Adversarial machine learning, Sensor attack, Autonomous driving

ACM Reference Format:
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 *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3319535.3339815

1 INTRODUCTION

Autonomous vehicles, or self-driving cars, are under rapid development, with some vehicles already found on public roads [9, 12, 14] In AV systems, one fundamental pillar is *perception*, which leverages sensors like cameras and LiDARs (Light Detection and Ranging) to understand the surrounding driving environment. Since such function is directly related to safety-critical driving decisions such as collision avoidance, multiple prior research efforts have been made to study the security of camera-based perception in AV settings. For example, prior work has reported sensor-level attacks such as camera blinding [42], physical-world camera attacks such as adding stickers to traffic signs [28, 29], and trojan attacks on the neural networks for AV camera input [37].

Despite the research efforts in camera-based perception, there is no thorough exploration into the security of LiDAR-based perception in AV settings. LiDARs, which measure distances to surrounding obstacles using infrared lasers, can provide 360-degree viewing angles and generate 3-dimensional representations of the road environment instead of just 2-dimensional images for cameras. Thus, they are generally considered as more important sensors than cameras for AV driving safety [3, 13] and are adopted by nearly all AV makers today [4, 5, 7, 11]. A few recent works demonstrated the feasibility of injecting spoofed points into the sensor input from the LiDAR [42, 44]. Since such input also needs to be processed by an object detection step in the AV perception pipeline, it is largely unclear whether such spoofing can directly lead to semantically-impactful security consequences, e.g., adding spoofed road obstacles, in the LiDAR-based perception in AV systems.

In this work, we perform the first study to explore the security of LiDAR-based perception in AV settings. To perform the analysis, we target the LiDAR-based perception implementation in Baidu Apollo, an open-source AV system that has over 100 partners and has reached a mass production agreement with multiple partners such as Volvo and Ford [8, 9]. We consider a LiDAR spoofing attack, i.e., injecting spoofed LiDAR data points by shooting lasers, as our threat model since it has demonstrated feasibility in previous work [42, 44]. With this threat model, we set the attack goal as adding spoofed obstacles in close distances to the front of a victim AV (or *front-near* obstacles) in order to alter its driving decisions.

In our study, we first reproduce the LiDAR spoofing attack from the work done by Shin et al. [44] and try to exploit Baidu Apollo's LiDAR-based perception pipeline, which leverages machine learning for object detection as with the majority of the state-of-the-art LiDAR-based AV perception techniques [6]. We enumerate different spoofing patterns from the previous work, e.g., a spoofed wall, and different spoofing angles and shapes, but none of them succeed in generating a spoofed road obstacle after the machine learning step. We find that a potential reason is that the current spoofing technique can only cover a very narrow spoofing angle, i.e., 8° horizontally in our experiments, which is not enough to generate a point cloud of a road obstacle near the front of a vehicle. Thus, blindly applying existing spoofing techniques cannot easily succeed.

To achieve the attack goal with existing spoofing techniques, we explore the possibility of strategically controlling the spoofed points to fool the machine learning model in the object detection step. While it is known that machine learning output can be maliciously altered by carefully-crafted perturbations to the input [18, 20, 29, 41, 57], no prior work studied LiDAR-based object detection models for AV systems. To approach this problem, we formulate the attack task as an optimization problem, which has been shown to be effective in previous machine learning security studies [21, 24, 26, 50, 51, 53]. Specific to our study, two functions need to be newly formulated: (1) an input perturbation function that models LiDAR spoofing capability in changing machine learning model input, and (2) an objective function that can reflect the attack goal. For the former, since previous work did not perform detailed measurements for the purpose of such modeling, we experimentally explore the capability of controlling the spoofed data points, e.g., the number of points and their positions. Next, we design a set of global spatial transformation functions to model these observed attack capabilities at the model input level. In this step, both the quantified attack capabilities and the modeling methodology are useful for future security studies of LiDAR-related machine learning models.

For the attack goal of adding front-near obstacles, designing a objective function is also non-trivial since the machine learning model output is post-processed in the perception module of Baidu Apollo before it is converted to a list of perceived obstacles. To address this, we study the post-processing logic, extract key strategies of transforming model output into perceived obstacles, and formulate it into the objective function.

With the optimization problem mathematically formulated, we start by directly solving it using optimization algorithms like previous studies [21]. However, we find that the average success rate of adding front-near obstacles is only 30%. We find that this is actually caused by the nature of the problem, which makes it easy for any optimization algorithm to get trapped in local extrema. To solve this problem, we design an algorithm that combines global sampling and optimization, which is able to successfully increase the average success rates to around 75%.

As a case study for understanding the impact of the discovered attack input at the AV driving decision level, we construct two attack scenarios: (1) *emergency brake attack*, which may force a moving AV to suddenly brake and thus injure the passengers or cause rear-end collisions, and (2) *AV freezing attack*, which may cause an AV waiting for the red light to be permanently "frozen" in the intersection and block traffic. Using real-world AV driving data traces released by the Baidu Apollo team, both attacks successfully trigger the attacker-desired driving decisions in Apollo's simulator.

Based on the insights from our security analysis, we propose defense solutions not only at AV system level, e.g., filtering out LiDAR data points from ground reflection, but also at sensor and machine learning model levels.

In summary, this work makes the following contributions:

- We perform the first security study of LiDAR-based perception for AV systems. We find that blindly applying existing LiDAR spoofing techniques cannot easily succeed in generating semantically-impactful security consequences after the machine learning-based object detection step. To achieve the attack goal with existing spoofing techniques, we then explore the possibility of strategically controlling the spoofed points to fool the machine learning model, and formulate the attack as an optimization problem.
- To perform analysis for the machine learning model used in LiDAR-based AV perception, we make two methodology-level contributions. First, we conduct experiments to analyze the LiDAR spoofing attack capability and design a global spatial transformation based method to model such capability in mathematical forms. Second, we identify inherent limitations of directly solving our problem using optimization, and design an algorithm that combines optimization and global sampling. This is able to increase the attack success rates to around 75%.
- As a case study to understand the impact of the attacks at the AV driving decision level, we construct two potential attack scenarios: emergency brake attack, which may hurt the passengers or cause a rear-end collision, and AV freezing attack, which may block traffic. Using a simulation based evaluation on real-world AV driving data, both attacks successfully trigger the attacker-desired driving decisions. Based on the insights, we discuss defense directions at AV system, sensor, and machine learning model levels.

2 BACKGROUND

2.1 LiDAR-based Perception in AV Systems

AVs rely on various sensors to perform real-time positioning (also called localization) and environment perception (or simply perception). LiDAR, camera, radar, and GPS/IMU are major sensors used by various autonomous driving systems. The data collected from those sensors are transformed and processed before it becomes useful information for AV systems. Fig. 1 shows the data processing pipeline of LiDAR sensor data in the perception module of Baidu Apollo [4]. As shown, it involves three main steps as follows:

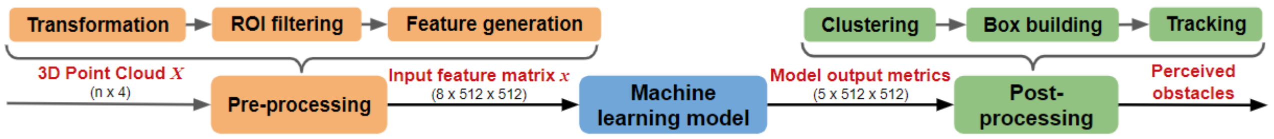


Figure 1: Overview of the data processing pipeline for LiDAR-based perception in Baidu Apollo.

Step 1: Pre processing. The raw LiDAR sensor input is called *3D point cloud* and we denote it as X. The dimension of X is $n \times 4$, where n denotes the number of data points and each data point is a 4-dimension vector with the 3D coordinates, $\mathbf{w_x}$, $\mathbf{w_y}$, and $\mathbf{w_z}$, and the intensity of the point. In the pre-processing step, X is first transformed into an absolute coordinate system. Next, the *Region of Interest (ROI)* filter removes unrelated portions of the 3D point cloud data, e.g., those that are outside of the road, based on HDMap information. Next, a *feature generation* process generates a feature matrix x ($8 \times 512 \times 512$), which is the input to the subsequent machine learning model. In this process, the ROI-filtered 3D point cloud within the range (60 meters by default) is mapped to 512×512 cells according to the $\mathbf{w_x}$ and $\mathbf{w_y}$ coordinates. In each cell, the assigned points are used to generate 8 features as listed in Table 1.

Step 2: DNN-based object detection. A Deep Neural Network (DNN) then takes the feature matrix x as input and produces a set of output metrics for each cell, e.g., the probability of the cell being a part of an obstacle. These output metrics are listed in Table 2.

Step 3: Post processing. The clustering process only considers cells with *objectness* values (one of the output metrics listed in Table 2) greater than a given threshold (0.5 by default). Then, the process constructs candidate object clusters by building a connected graph using the cells' output metrics. Candidate object clusters are then filtered by selecting clusters with average *positiveness* values (another output metric) greater than a given threshold (0.1 by default). The *box builder* then reconstructs the bounding box including height, width, length of an obstacle candidate from the 3D point cloud assigned to it. Finally, the *tracker* integrates consecutive frames of processed results to generate tracked obstacles, augmented with additional information such as speed, acceleration, and turning rates, as the output of the LiDAR-based perception.

With the information of perceived obstacles such as their positions, shapes, and obstacle types, the Apollo system then uses such information to make driving decisions. The perception output is further processed by the *prediction* module which predicts the future trajectories of perceived obstacles, and then the *planning* module which plans the future driving routes and makes decisions such as stopping, lane changing, yielding, etc.

Feature	Description
Max height	Maximum height of points in the cell.
Max intensity	Intensity of the brightest point in the cell.
Mean height	Mean height of points in the cell.
Mean intensity	Mean intensity of points in the cell.
Count	Number of points in the cell.
Direction	Angle of the cell's center with respect to the origin.
Distance	Distance between the cell's center and the origin.
Non-empty	Binary value indicating whether the cell is empty or occupied.

Table 1: DNN model input features.

Metrics	Description
Center offset	Offset to the predicted center of the cluster the cell belongs to.
Objectness	The probability of a cell belonging to an obstacle.
Positiveness	The confidence score of the detection.
Object height	The predicted object height.
Class probability	The probability of the cell being a part of a vehicle, pedestrian, etc.

Table 2: DNN model output metrics.

2.2 LiDAR Sensor and Spoofing Attacks

To understand the principles underlying our security analysis methodology, it is necessary to understand how the LiDAR sensor generates a point cloud and how it is possible to alter it in a controlled way using spoofing attacks.

LiDAR sensor. A LiDAR sensor functions by firing laser pulses and capturing their reflections using photodiodes. Because the speed of light is constant, the time it takes for the echo pulses to reach the receiving photodiode provides an accurate measurement of the distance between a LiDAR and a potential obstacle. By firing the laser pulses at many vertical and horizontal angles, a LiDAR generates a point cloud used by the AV systems to detect objects.

LiDAR spoofing attack. Sensor spoofing attacks use the same physical channels as the targeted sensor to manipulate the sensor readings. This strategy makes it very difficult for the sensor system to recognize such attack, since the attack doesn't require any physical contact or tampering with the sensor, and it doesn't interfere with the processing and transmission of the digital sensor measurement. These types of attack could trick the victim sensor to provide seemingly legitimate but actually erroneous data.

LiDAR has been shown to be vulnerable to laser spoofing attacks in prior work. Petit et al. demonstrated that a LiDAR spoofing attack can be performed by replaying the LiDAR laser pulses from a different position to create fake points further than the location of the spoofer [42]. Shin et al. showed that it is possible to generate a fake point cloud at different distances, even closer than the spoofer location [44]. In this paper, we build upon these prior work to study the effect of this attack vector on the security of AV perception.

2.3 Adversarial Machine Learning

Neural networks. A neural network is a function consisting of connected units called (artificial) neurons that work together to represent a differentiable function that outputs a distribution. A given neural network (e.g., classification) can be defined by its model architecture and parameters ϕ. An optimizer such as Adam [35] is used to update the parameters ϕ with respect to the objective function $\mathcal{L}$.

Adversarial examples. Given a machine learning model M, input x and its corresponding label y, an adversarial attacker aims to generate adversarial examples x' so that $M(x') \neq y$ (untargeted attack) or $M(x') = y'$, where y' is a target label (targeted attack). Carlini and Wagner [21] proposed to generate an adversarial perturbation for a targeted attack by optimizing an objective function as follows:

$$\min ||x - x'||_p \quad \text{s.t.} \quad M(x') = y' \quad \text{and} \quad x' \in X,$$

where $M(x') = y'$ is the target adversarial goal and $x' \in X$ denote that the adversarial examples should be in a valid set. Further, optimization-based algorithms have been leveraged to generate adversarial examples on various kinds of machine learning tasks successfully, such as segmentation [26, 53], human pose estimation [26], object detection [53], Visual Question Answer system [54], image caption translation [24], etc. In this paper, we also leverage an optimization-based method to generate adversarial examples to fool LiDAR-based AV perception.

3 ATTACK GOAL AND THREAT MODEL

Attack goal. To cause semantically-impactful security consequence in AV settings, we set the attack goal as fooling the LiDAR-based perception into perceiving fake obstacles in front of a victim AV in order to maliciously alter its driving decisions. More specifically, in this work, we target *front-near* fake obstacles, i.e., those that are in close distances to the front of a victim AV, since they have the highest potential to trigger immediate erroneous AV driving decisions. In this work, we define front-near obstacles as those that are around 5 meters to the front of a victim AV.

Threat model. To achieve the attack goal above, we consider LiDAR spoofing attacks as our threat model, which is a demonstrated practical attack vector for LiDAR sensors [42, 44] as described in §2.2. In AV settings, there are several possible scenarios to perform such attack. First, the attacker can place an attacking device at the roadside to shoot malicious laser pulses to AVs passing by. Second, the attacker can drive an attack vehicle in close proximity to the victim AV, e.g., in the same lane or adjacent lanes. To perform the attack, the attack vehicle is equipped with an attacking device that shoots laser pulses to the victim AV's LiDAR. To perform laser aiming in these scenarios, the attacker can use techniques such as camera-based object detection and tracking. In AV settings, these attacks are stealthy since the laser pulses are invisible and laser shooting devices are relatively small in size.

As a first security analysis, we assume that the attacker has white-box access to the machine learning model and the perception system. We consider this threat model reasonable since the attacker could obtain white-box access by additional engineering efforts to reverse engineering the software.

4 LIMITATION OF BLIND SENSOR SPOOFING

To understand the security of LiDAR-based perception under LiDAR spoofing attacks, we first reproduce the state-of-the-art LiDAR spoofing attack by Shin et al. [44], and explore the effectiveness of directly applying it to attack the LiDAR-based perception pipeline in Baidu Apollo [4], an open-source AV system that has over 100 partners and has reached mass production agreement with multiple partners such as Volvo, Ford, and King Long [8, 9].

Spoofing attack description. The attack by Shin et al. [44] consists of three components: a photodiode, a delay component, and an infrared laser, which are shown in Fig. 3. The photodiode is used to synchronize with the victim LiDAR. The photodiode triggers the delay component whenever it captures laser pulses fired from the victim LiDAR. Then the delay component triggers the attack laser after a certain amount of time to attack the following firing cycles of the victim LiDAR. Since the firing sequence of laser pulses is consistent, an adversary can choose which fake points will appear in the point cloud by crafting a pulse waveform to trigger the attack laser.

Experimental setup. We perform spoofing attack experiments on a VLP-16 PUCK LiDAR System from Velodyne [32]. The VLP-16 uses a vertical array of 16 separate laser diodes to fire laser pulses at different angles. It has a 30 degree vertical angle range from -15 ° to +15 °, with 2 ° of angular resolution. The VLP-16 rotates horizontally around a center axis to send pulses in a 360 ° horizontal range, with a varying azimuth resolution between 0.1 ° and 0.4 °. The laser firing sequence follows the pattern shown in Figure 4. The VLP-16 fires 16 laser pulses in a cycle every 55.296 μs, with a period of 2.304 μs. The receiving time window is about 667 ns. We chose this sensor because it is compatible with Baidu Apollo and uses the same design principle as the more advanced HDL-64E LiDARs used in many AVs. The similar design indicates that the same laser attacks that affect the VLP-16 can be extended to high-resolution LiDARs like the HDL-64E.

We use the OSRAM SFH 213 FA as our photodiode, with a comparator circuit similar to the one used by Shin et al. We use a Tektronix AFG3251 function generator as the delay component with the photodiode circuit as an external trigger. In turn, the function generator provides the trigger to the laser driver module PCO-7114 that drives the attack laser diode OSRAM SPL PL90. With the PCO-7114 laser driver, we were able to fire the laser pulses at the same pulse rate of the VLP-16, 2.304 μs, compared to 100 μs of the previous work. An optical lens with a diameter of 30mm and a focal length of 100 mm was used to focus the beam, making it more effective for ranges farther than 5 meters. We generate the custom pulse waveform using the Tektronix software ArbExpress [2] to create different shapes and the Velodyne software VeloView [10] to analyze and extract the point clouds.

Experiment results. The prior work of Shin et al. is able to spoof a maximum of 10 fake dots in a single horizontal line. With our setup improvements (a faster firing rate and a lens to focus the beam), fake points can be generated at all of the 16 vertical viewing angles and an 8 ° horizontal angle at greater than 10 meters away. In total, around 100 dots can be spoofed by covering these horizontal and vertical angles (illustrated in Fig. 14 in Appendix). These spoofed dots can also be shaped by modifying the custom pulse waveform used to fire the attack laser. Noticed that even though around 100 dots can be spoofed, they are not all spoofed stably. The attacker is able to spoof points at different angles because the spoofed laser pulses hit a certain area on the victim LiDAR due to the optical lens focusing. The closer to the center of the area, the stronger and stabler laser pulses are received by the victim LiDAR. We find that among 60 points at the center 8-10 vertical lines can be stably spoofed with high intensity.

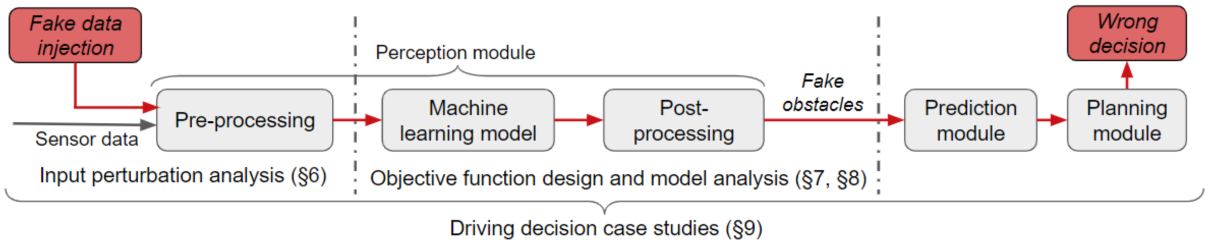


Figure 2: Overview of the Adv-LiDAR methodology.

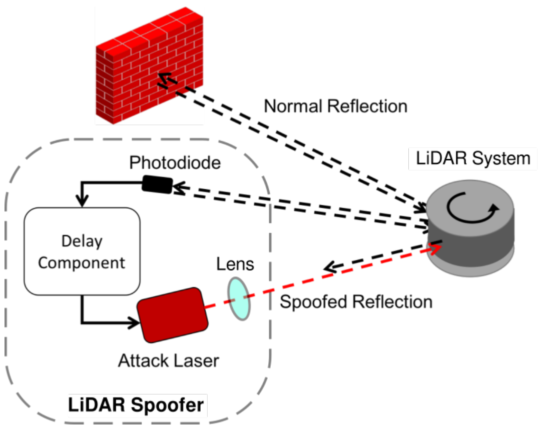


Figure 3: Illustration of LiDAR spoofing attack. The photodiode receives the laser pulses from the LiDAR and activate the delay component that triggers the attacker laser to simulate real echo pulses.

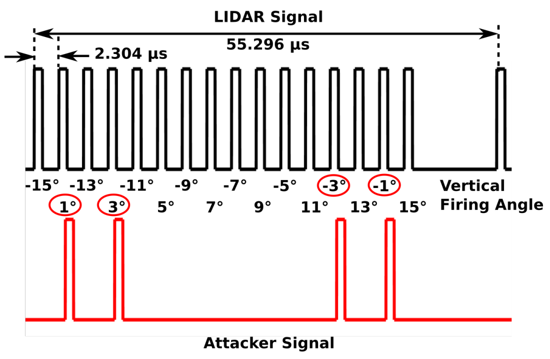


Figure 4: The consistent firing sequence of the LiDAR allows an attacker to choose the angles and distances from which spoofed points appear. For example, applying the attacker signal, fake dots will appear at 1°, 3°, -3°, and -1° angles (0° is the center of the LiDAR)

4.1 Blind LiDAR Spoofing Experiments

After reproducing the LiDAR spoofing attack, we then explore whether blindly applying such attack can directly generate spoofed obstacles in the LiDAR-based perception in Baidu Apollo. Since our LiDAR spoofing experiments are performed in indoor environments,

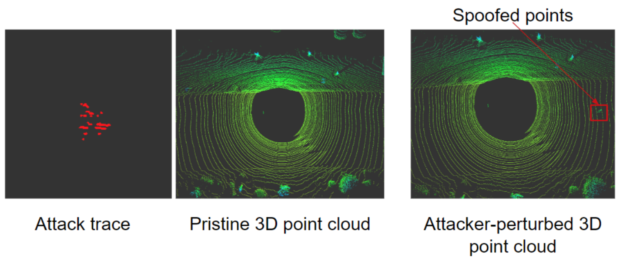


Figure 5: Generating the attacker-perturbed 3D point cloud by synthesizing the pristine 3D point cloud with the attack trace to spoof a front-near obstacle 5 meters away from the victim AV.

we synthesize the on-road attack effect by adding spoofed LiDAR points to the original 3D point cloud collected by Baidu Apollo team on local roads at Sunnyvale, CA. The synthesizing process is illustrated in Fig. 5. After this process, we run Apollo's perception module with the attacker-perturbed 3D point cloud as input to obtain the object detection output. In this analysis, we explore three blind attack experiments as follows:

Experiment 1: Directly apply original spoofing attack traces. In this experiment, we directly replay spoofing attack traces to attack LiDAR-based perception in Apollo. More specifically, we experiment with attack traces obtained from two sources: (1) the original spoofing attack traces from Shin et al. [44], and (2) the attack traces generated from the spoofing attack reproduced by us, which can inject more dots after our setup improvements. However, we are not able to observe a spoofed obstacle for any of these traces at the output of the LiDAR-based perception pipeline.

Experiment 2: Apply spoofing attack traces at different angles. To understand whether successfully spoofing an obstacle depends on the angle of the spoofed points, in this experiment we inject spoofed points at different locations. More specifically, we uniformly sample 100 different angles out of 360 degrees around the victim AV, and inject the spoofing attack traces reproduced by us. However, we are not able to observe spoofed obstacles for any of these angles.

Experiment 3: Apply spoofing attack traces with different shapes. To understand whether successfully spoofing an obstacle depends on the pattern of the spoofed points, in this experiment we inject points with different spoofing patterns. More specifically, we generate random patterns of spoofed points by randomly setting

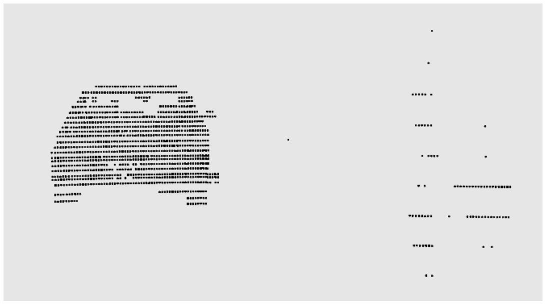


Figure 6: The point cloud from a real vehicle reflection (left) and from the spoofing attack (right) in a 64-line HDL-64E LiDAR. The vehicle is around 7 meters in front of the AV.

distances for each point at different angles. We generate 160 points covering 16 vertical lines, 10 points for each line with continuous horizontal angles. To trigger immediate control decision changes in an AV, the spoofed obstacle needs to be close to the victim AV. Thus, we set the generated distances of the spoofed point to be within 4 to 6 meters to the victim AV. We generate 100 different spoofed patterns in total, but we are not able to observe spoofed obstacles for any of these patterns.

Summary. In these experiments, we try various blind spoofing attack strategies directly derived from the state-of-the-art LiDAR spoofing attack, but none of them succeed in generating spoofed obstacles in the LiDAR-based perception pipeline in Baidu Apollo. There are two potential reasons. First, as described earlier, the current attack methodology can only cover a very narrow spoofing angle, i.e., 8 ° of horizontal angle even after our setup improvements. Second, the coverage of vertical angles is limited by the frequency of spoofing laser pulses. Thus, when attacking a LiDAR with more vertical angles, e.g., a 64-line LiDAR, since a 64-line LiDAR takes similar time as a 16-line LiDAR in scanning vertical angles, the attacker cannot spoof more vertical angles than those for a 16-line LiDAR. Thus, the current methodology limits the number of spoofed points, making it hard to generate enough points to mimic an important road obstacle.

To illustrate that, as shown in Fig. 6, the point cloud for a real vehicle has a much wider angle and much more points than the attack traces reproduced by us. Thus, blindly applying the spoofing attack cannot easily fool the machine learning based object detection process in the LiDAR-based perception pipeline. In the next section, we explore the possibility of further exploiting machine learning model vulnerabilities to achieve our attack goal.

5 IMPROVED METHODOLOGY: ADV-LIDAR

As discussed in §4, without considering the machine learning model used in LiDAR-based perception, blindly applying existing LiDAR spoofing attacks can hardly achieve the attack goal of generating front-near obstacles. Since it is known that machine learning output can be maliciously altered by carefully-crafted perturbations to the input [18, 20, 29, 41, 57], we are then motivated to explore the possibility of strategically controlling the spoofed points to fool the machine learning model in LiDAR-based perception. In this section, we first describe the technical challenges after involving adversarial machine learning analysis in this research problem, and then present our solution methodology overview, called *Adv-LiDAR*.

5.1 Technical Challenges

Even though previous studies have shown promising results in attacking machine learning models, none of them studied LiDAR-based object detection models, and their approaches have limited applicability to our analysis goal due to three challenges:

First, attackers have limited capability of perturbing machine learning model inputs in our problem. Other than perturbing pixels on an image, perturbing machine learning inputs under AV settings requires perturbing 3D point cloud raw data by sensor attack and bypassing the associated pre-processing process. Therefore, such perturbation capability needs to be quantified and modeled.

Second, optimization-based methods for generating adversarial examples in previous studies may not be directly suitable for our analysis problem due to the limited model input perturbation capability. As shown in §7, we find that optimization-based methods are inherently limited due to the nature of our problem, and can only achieve very low success rate in generating front-near obstacles.

Third, in our problem, successfully changing the machine learning model output does not directly lead to successes in achieving our attack goal in AV settings. As detailed later in §7, in AV systems such as Baidu Apollo, machine learning model output is post-processed before it is converted to a list of perceived obstacles. Thus, an objective function that can effectively reflect our attack goal needs to be newly designed.

5.2 Adv-LiDAR Methodology Overview

In this section, we provide an overview of our solution methodology, which we call Adv-LiDAR, that addresses the three challenges above. At a high level, to identify adversarial examples for the machine learning model M, we adopt an optimization-based approach, which has shown both high efficiency and effectiveness by previous studies for machine learning models across different domains [21, 26, 51, 52]. To help explain the formulation of the optimization problem, we summarize the notations in Table 3. Specifically, the problem is formulated as follows:

$$\begin{aligned} \min \quad & \mathcal{L}_{\text{adv}}(x \oplus t'; M) \\ \text{s.t.} \quad & t' \in \{\Phi(T') | T' \in \mathcal{A}\} \ \ \& \ \ x = \Phi(X) \end{aligned} \tag{1}$$

where X is the pristine 3D point cloud and x represents the corresponding 2D input feature matrix. $\Phi(\cdot)$ is the pre-processing function that maps X into x (§2.1). T' and t' are the corresponding adversarial spoofed 3D point cloud and adversarial spoofed input feature matrix. $\mathcal{A}$ is a set of spoofed 3D point cloud generated from LiDAR spoofing attacks. $\mathcal{L}_{\text{adv}}(\cdot; M)$ is the adversarial loss designed to achieve the adversarial goal given the machine learning model M. The constraints are used to guarantee that the generated adversarial examples t' satisfy the spoofing attack capability.

Figure 2 overviews the analysis tasks needed to solve the optimization problem. First, we need to conduct an input perturbation analysis that formulates the spoofing attack capabilities $\mathcal{A}$ and merging function $\oplus$. Second, we need to perform a model analysis to design an objective function to generate adversarial examples. Third, as a case study to understand the impact of the attacks at the

Notation	Description	Notation	Description
X	3D point cloud	x	Input feature matrix
X'	Adversarial 3D point cloud	x'	Adversarial input feature matrix
T	Spoofed 3D point cloud	t	Spoofed input feature matrix
T'	Adversarial spoofed 3D point cloud	t'	Adversarial spoofed input feature matrix
$(\mathbf{w_x}, \mathbf{w_y}, \mathbf{w_x})$	3D Cartesian coordinate	L_θ, L_τ	Upper bound of θ, τ during sampling
(u, v)	Coordinate of t	(u', v')	Coordinate of t'
M	Machine learning model	$I.$	Model outputs
$N(u, v)$	4-pixel neighbor at the location (u, v)	$S(\cdot)$	Height Scaling function
$\mathcal{A}$	Spoofing attack capability	$\Phi(\cdot)$	Mapping function (3D$\longrightarrow$ 2D)
$Q(M, \cdot)$	Extraction function	$\oplus(\cdot)$	Merge function
$\mathcal{M}(\cdot)$	Gaussian mask	(px, py)	Center points of the Gaussian mask
$f(\cdot)$	Objective function	$\mathcal{L}_{\text{adv}}(\cdot)$	Adversarial loss
$H(\theta, \tau, \epsilon)$	2D Homography Matrix (θ : rotation, ϵ : scaling ; τ : translation)	S_h	Height scaling ratio
$\mathcal{S}_{\text{T}}$	Set of spoofed 3D point cloud	$\mathcal{S}_{\text{t}}$	Set of spoofed input feature matrix
$G_T(T, \cdot)$	Global spatial transformation function for 3D point cloud	$G_t(t, \cdot)$	Global spatial transformation function for input feature matrix

Table 3: Notations adopted in this work.

AV driving decision level, we further perform a driving decision analysis using the identified adversarial examples. More details about these tasks are as follows:

Input perturbation analysis. Formulating $\mathcal{A}$ and $\oplus$ is non-trivial. First, previous work regarding LiDAR spoofing attacks neither provided detailed measurements on the attacker's capability in perturbing 3D point cloud nor expressed it in a closed form expression. Second, point cloud data is pre-processed by several steps as shown in Section 2.1 before turning into machine learning input, which means the merging function $\oplus$ cannot be directly expressed. To address these two challenges, as will be detailed later in §6, we first conduct spoofing attacks on LiDAR to collect a set of possible spoofed 3D point cloud. Using such spoofed 3D point cloud, we model the spoofing attack capability $\mathcal{A}$. We further analyze the pre-processing program to obtain the additional constraints to the machine learning input perturbation, or the spoofed input feature matrix. Based on this analysis, we formulate the spoofed input feature matrix into a differentiable function using global spatial transformations, which is required for the model analysis.

Objective function design and model analysis. As introduced earlier in §5.1, in LiDAR-based perception in AV systems, the machine learning model output is post-processed (§ 2.1) before turning into a list of perceived obstacles. To find an effective objective function, we study the post-processing steps to extract key strategies of transforming model output into perceived obstacles, and formulate it into an objective function that reflects the attack goal. In addition, we find that our optimization problem cannot be effectively solved by directly using existing optimization-based methods. We analyze the loss surface, and find that this inefficiency is caused by the problem nature. To address this challenge, we improve the methodology by combining global sampling with optimization. Details about the analysis methodology and results are in §7 and § 8.

Driving decision case study. With the results from previous analysis steps, we can generate adversarial 3D point cloud that can inject spoofed obstacles at the LiDAR-based perception level. To understand their impact at the AV driving decision level, we construct and evaluate two attack scenarios as case studies. The evaluation methodology and results are detailed later in §9.

6 INPUT PERTURBATION ANALYSIS

To generate adversarial examples by solving the above optimization problem in Equation 1, we need to formulate merging function $\oplus$ and input feature matrix spoofing capability $\Phi(\mathcal{A})$ as a closed form. In this section, we first analyze the spoofing attack capability ($\mathcal{A}$), and then use it to formulate $\Phi(\mathcal{A})$.

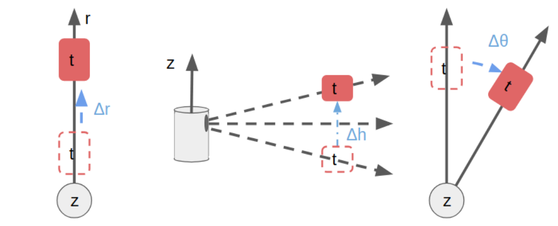
6.1 Spoofing Attack Capability

Based on the attack reproduction experiments in §4, the observed attack capability ($\mathcal{A}$) can be described from two aspects:

Number of spoofed points. As described in §4, even though it is possible to spoof around 100 points after our setup improvement, we find that around 60 points can be reliably spoofed in our experiments. Thus, we consider 60 as the highest number of reliable spoofed points. Noticed that, the maximum number of spoofed points could be increased if the attacker uses more advanced attack equipment. Here, we choose a set of devices that are more accessible (detailed in §4) and end up with the ability to reliably spoof around 60 points. In addition, considering that an attacker may use a slower laser or cruder focusing optics, such as in the setup by Shin et al. [44], we also consider 20 and 40 spoofed points in our analysis.

Location of spoofed points. Given the number of spoofed points, the observed attack capability in placing these points are described and modeled as follows:

(1) Modify the *distance* of the spoofed point from the LiDAR by changing the delay of the attack laser signal pulses in small intervals (nanosecond scale). From the perspective of spoofed 3D point cloud T, this can be modeled as moving the position of the spoofed points nearer or further on the axis r that connects the spoofed points and the LiDAR sensor by distance Δr (Fig. 7 (a)).
(2) Modify the *altitude* of a spoofed point within the vertical range of the LiDAR by changing the delay in intervals of 2.304 μs. From the perspective of spoofed 3D point cloud T, this can be modeled as moving the position of the spoofed points from vertical line to vertical line to change the height of it by height Δh (Fig. 7 (b)).
(3) Modify the *azimuth* of a spoofed point within a horizontal viewing angle of 8° by changing the delay in intervals of 55.296 μs. By moving the LiDAR spoofer to different locations



(a) Distance (b) Altitude (c) Azimuth

Figure 7: Attack capability in perturbing 3D Point Cloud T

around the LiDAR, it is possible to spoof at any horizontal angle. From the perspective of spoofed 3D point cloud T, this can be modeled as rotating the spoofed points with the LiDAR sensor as the pivot point on the horizontal plane by angle $\Delta\theta$ (Fig. 7 (c)).

Therefore, we model the attack capability $\mathcal{A}$ by applying these three modifications to the given spoofed 3D point cloud T. Here the spoofed 3D point cloud is collected by reproducing the sensor spoofing attack. The point number of T can be 20, 40 and 60 to represent different attack capabilities as mentioned before. In the next section, the attack capability $\mathcal{A}$ modeled here is used to model the perturbation of the input feature matrix x.

6.2 Input Perturbation Modeling

After analyzing spoofing attack capability $\mathcal{A}$, to formulate $x \oplus t'$ in Equation 1, We need to have the following steps: (1) formulating the merging function $\oplus$; (2) modeling the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ based on known spoofing attack capability $\mathcal{A}$. In this section, we first formulate the merging function $\oplus$ by analyzing the pre-processing program. Then we model the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ by expressing t' with spoofed input feature matrix t in a differentiable function using global spatial transformations. Here, spoofed input feature matrix t can be attained with a given spoofed 3D point cloud T by $t = \Phi(T)$.

Formulating merging function ($\oplus$). To model the merging function $\oplus$ operated on x and t', which are in the domain of input feature matrix, we need to first analyze the pre-processing program $\Phi(\cdot)$ that transforms the 3D point cloud X into the input feature matrix x. As described in §2.1, the pre-processing process consists of three sub-processes: *coordinate transformation*, *ROI filtering* and *input feature matrix extraction*. The first two processes make minor effects on the adversarial spoofed 3D point cloud T' generated by the spoofing attack we conducted in §6. The *coordinate transformation* process has no effect because the adversarial spoofed 3D point cloud T' will be transformed along with the 3D point cloud X. As for the *ROI filtering* process, it filters out 3D point cloud located outside of the road from a bird's-eye view. Therefore, as long as we spoof points on the road, the *ROI filtering process* makes no effect on the adversarial spoofed 3D point cloud T'. The *feature extraction* process, as we mentioned in Section 2.1, extracts statistical features such as average height (I_{avg_h}), average intensity (I_{avg_int}), max height (I_{max_h}) and so on.

Because of such pre-processing, the spoofed input feature matrix t' cannot be directly added to the input feature matrix x to attain the adversarial input feature matrix x'. To attain x', we express such "addition" operation ($\oplus$) as a differentiable function shown below. Note that in this equation we do not include a few features in Table 1 such as direction and distance since they are either constant or can be derived directly from the features included in the equation.

$$x' = x \bigoplus t' = \begin{bmatrix} I^x_{cnt} + I^{t'}_{cnt} \\ (I^x_{avg\_h} \cdot I^x_{cnt} + I^{t'}_{avg\_h} \cdot I^{t'}_{cnt})/(I^x_{cnt} + I^{t'}_{cnt}) \\ \max(I^x_{max\_h}, I^{t'}_{max\_h}) \\ (I^x_{avg\_int} \cdot I^x_{cnt} + I^{t'}_{avg\_int} \cdot I^{t'}_{cnt})/(I^x_{cnt} + I^{t'}_{cnt}) \\ \sum I^x_{max\_int} \cdot 1\{I^x_{max\_h} = max\{I^x_{max\_h}, I^{t'}_{max\_h}\}\} \end{bmatrix} \tag{2}$$

Modeling input feature matrix spoofing capability $\Phi(\mathcal{A})$. To model input feature matrix spoofing capability $\Phi(\mathcal{A})$, it equals to representing adversarial input feature matrix t' with known spoofed input feature matrix t. We can use global spatial transformations including rotation, translation and scaling, under certain constraints to represent the input feature matrix spoofing capability. Here the translation and scaling transformation interprets the attack capability in terms of modifying the *azimuth* of 3D point cloud while the rotation transformation interprets the attack capability in terms of modifying the *distance* of 3D point cloud from the LiDAR.

Specifically, we apply the global spatial transformation to a set of the spoofed input feature matrix $\mathcal{S}_t$ to formulate the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ and to represent adversarial spoofed input feature matrix t'. For each spoofed input feature matrix $t \in \mathcal{S}_t$, it is mapped from a corresponding spoofed 3D point cloud T such that $t = \Phi(T)$.

We use $t'_{(i)}$ to denote values of the i-th position on the spoofed input feature matrix t' and 2D coordinate $(u'_{(i)}, v'_{(i)})$ to denote its location. t' is transformed from an arbitrary instance t where $t \in \mathcal{S}_t$ by applying a homography matrix $H(\theta, \tau, \epsilon)$. The location of $t_{(i)}$ can be derived as $t'_{(i)}$ as follows:

$$\begin{aligned} & (u_{(i)}, v_{(i)}, 1)^T = H \cdot (u'_{(i)}, v'_{(i)}, 1)^T, \\ \text{w.r.t.} \quad & H = \begin{bmatrix} \epsilon(\cos\theta & -\sin\theta) & \tau_x \\ \epsilon(\sin\theta & \cos\theta) & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3}$$

Notice that here, τ_x/τ_y has a fixed ratio $\tan\theta$ since the translation is performed along the r axis shown in Fig. 7 (1). Since θ is dependent on the spoofed input feature matrix we provide for performing the transformation, we align the spoofed input feature matrix in advance to the x axis where $\theta = 0$ and accordingly $\tau_y = \tau_x \tan\theta = 0$. Therefore, we can optimize τ_x alone. Also, this process is equivalent to scaling so we remove ϵ.

We use the differentiable bilinear interpolation [34] to calculate $t'_{(i)}$:

$$t'_{(i)} = \sum_{q \in \mathcal{N}(u_{(i)}, v_{(i)})} t_{(q)}(1 - |u_{(i)} - u_{(q)}|)(1 - |v_{(i)} - v_{(q)}|), \tag{4}$$

where $\mathcal{N}(u_{(i)}, v_{(i)})$ represents the 4-pixel neighbors (top-left, top-right,bottom-left, bottom-right) at the location $(u(i), v(i))$.

Further, we can observe that the input feature matrix contains the height information as shown in Table 1. So we also optimize a global scale scalar s_h to the height features when generating adversarial spoofed input feature matrix t'. Define $S(t, s_h)$ as the scaling function that multiplies the features which contain the height information by s_h. Based on this transformation, Equation 4 will be changed as follows. For simplification, we denote the whole transformation progress as G_t. So $G_t(\theta, \tau_x, s_h; t)$ represents the transformed adversarial spoofed input feature matrixgiven spoofed input feature matrix t with transformation parameters θ, τ_x, s_h.

$$\begin{aligned} t'_{(i)} &= G_{t(i)}(\theta, \tau_x, s_h; t) \\ &= \sum_{q \in \mathcal{N}(u_{(i)}, v_{(i)})} S(t_{(q)}, s_h)(1 - |u_{(i)} - u_{(q)}|)(1 - |v_{(i)} - v_{(q)}|) \end{aligned} \tag{5}$$

7 GENERATING ADVERSARIAL EXAMPLES

After modeling the input perturbation, in this section we design the objective function with an effective adversarial loss $\mathcal{L}_{\text{adv}}$, and leverage an optimization method to find the attack transformation parameters that minimize such loss.

Design the adversarial loss $\mathcal{L}_{\text{adv}}$. Unlike previous work that performs the analysis only at the machine learning model level, there is no obvious objective function reflecting our attack goal of spoofing front-near obstacles. Yet, creating an effective objective function has been shown to be essential in generating effective adversarial examples [21]. In order to design an effective objective function, we analyze the post-processing step for the machine learning output. As shown in §2.1, in the clustering process, each cell of the model output is filtered by its *objectness* value. After the clustering process, candidate object clusters are filtered by their *positiveness* values. Upon such observation, we designed the adversarial loss $\mathcal{L}_{\text{adv}}$ as follows,

$$\mathcal{L}_{adv} = \sum(1 - Q(x', \text{positiveness})Q(x', \text{objectness}))\mathcal{M}(px, py) \tag{6}$$

where $Q(x', \cdot)$ is the function to extract the probabilities of $\cdot$ attribute from model M by feeding in adversarial example x'. $\mathcal{M}$ is a standard Gaussian mask with center coordinate (px, py) which is an attack target position chosen by the attacker. We attain (px, py) by mapping the attack target position in the real world onto the corresponding coordinates of the cell in the input feature matrix using Φ. The adversarial loss is then the summation over all the cells in the input feature matrix of the weighted value described above. By minimizing this designed adversarial loss, it equals to increasing the probability to detect the obstacle of the adversarial spoofed 3D point cloud given the machine learning model M.

Optimization algorithm and our improvement using sampling. With the $\mathcal{L}_{\text{adv}}$ design above, the optimization problem can be directly solved by using the Adam optimizer [35] to obtain the transformation parameters θ, τ_x and scalar s_h by minimizing the following objective function:

$$f = \arg\min_{\theta, \tau_x, s_h} \sum(1 - Q(x', \text{positiveness})Q(x', \text{objectness}))\mathcal{M}(px, py) \tag{7}$$

where t' can be obtained by Equation 5 and $x' = x \oplus t'$. In this paper, we call this direct solution *vanilla optimization.*

We visualize the loss surface against the transformation parameters in Fig. 9. During the vanilla optimization process, we observe that the loss surface over the transformation parameters is noisy at a small scale (green line) and quite flat at a large scale (red line). This leads to the problem of choosing a proper step size for optimization-based methods. For example, choosing a small step size will trap the optimizing process near a local minimum while choosing a large step size will be less effective due to noisy local loss pointing to the wrong direction. Different from Carlini et al. [21] that directly chose multiple starting points to reduces the trap of local minima, the optimization process under our setting is easy to get stuck in bad local minima due to the hard constraints of the perturbations. We propose a way to use *sampling* at a larger scale and to *optimize* at a smaller scale. To initiate the optimization process at different positions, we first calculate the range of the transformation parameters so that the transformed spoofed 3D point cloud is located in the target area. Then we uniformly take n samples for rotation and translation parameters and compose n^2 samples to initiate with.

Generating adversarial spoofed 3D point cloud. To further construct the adversarial 3D point cloud X', we need to construct adversarial spoofed 3D point cloud T'. Using the transformation parameters $\theta, \tau, \epsilon, s_h$, we can express the corresponding adversarial spoofed 3D point cloud T' such that $t' = \Phi(T')$ with a dual transformation function G_T of G_t. We use $T_{\mathbf{w}_\text{x}}, T_{\mathbf{w}_\text{y}}, T_{\mathbf{w}_\text{z}}$ to denote value of coordinate $(\mathbf{w}_\text{x}, \mathbf{w}_\text{y}, \mathbf{w}_\text{z})$ and T_i to denote the value of intensity for all points in spoofed 3D point cloud T. With transformation parameters $\theta, \tau, \epsilon, s_h$, we can express $T'_{\mathbf{w}_\text{x}}, T'_{\mathbf{w}_\text{y}}, T'_{\mathbf{w}_\text{z}}$ of the transformed adversarial spoofed 3D point cloud T' in Equation 8.

$$\begin{gathered} T'_i = T_i \\ \begin{bmatrix} T'_{\mathbf{w}_\text{x}} \\ T'_{\mathbf{w}_\text{y}} \\ T'_{\mathbf{w}_\text{z}} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & \tau_x \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & s_h & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} T_{\mathbf{w}_\text{x}} \\ T_{\mathbf{w}_\text{y}} \\ T_{\mathbf{w}_\text{z}} \\ 1 \end{bmatrix} \end{gathered} \tag{8}$$

Therefore, we can use $T' = G_T(\theta, \tau_x, s_h; T)$ represents the transformed adversarial spoofed 3D point cloud given spoofed 3D point cloud T with transformation parameters θ, τ_x, s_h.

Overall adversarial example generation process. Fig. 8 provides an overview of the overall adversarial example generation process. Given 3D point cloud X and spoofed 3D point cloud T (Fig. 8 (a)), we first map them via Φ to get corresponding input feature matrix x and spoofed input feature matrix t. Then we apply the sampling algorithm to initialize the transformation parameters θ, τ_x, s_h as shown in Fig. 8 (b). After the initialization, we leverage optimizer *opt* to further optimize the transformation parameters (θ, τ_x, s_h) with respect to the adversarial loss function $\mathcal{L}_{\text{adv}}$ (Fig. 8 (c)). With the transformation parameters θ, τ_x, s_h and T, we apply the dual transformation function G_T using the Equation 8 to get adversarial spoofed 3D point cloud T'. At last, to obtain the adversarial 3D point cloud X', we append T' to 3D point cloud X (Fig. 8 (d)). The entire adversarial example generation algorithm including the optimization parameters is detailed in Appendix A.

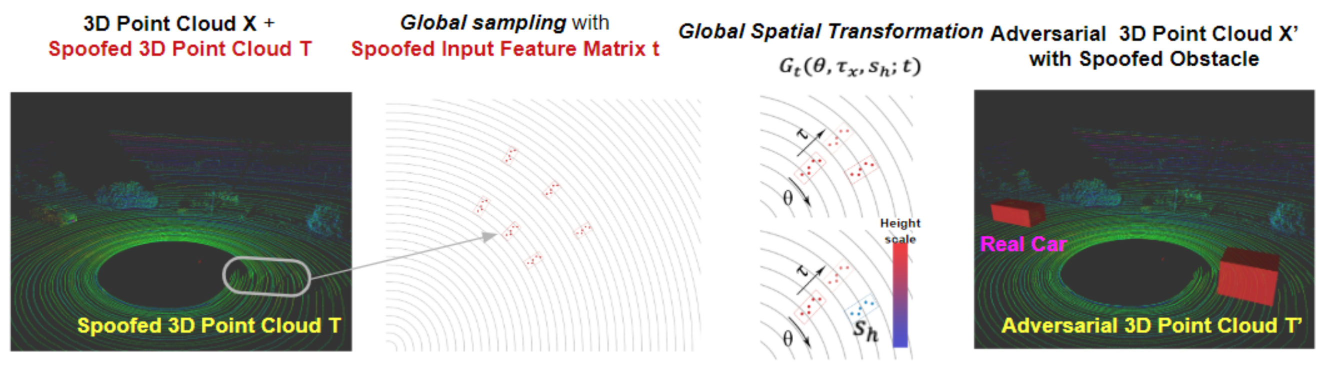

3D Point Cloud X +
Spoofed 3D Point Cloud T
Global sampling with
Spoofed Input Feature Matrix t
Global Spatial Transformation
$G_t(\theta, \tau_x, s_h; t)$
Adversarial 3D Point Cloud X'
with Spoofed Obstacle
Spoofed 3D Point Cloud T
Height scale
Real Car
Adversarial 3D Point Cloud T'

Figure 8: Overview of the adversarial example generation process.

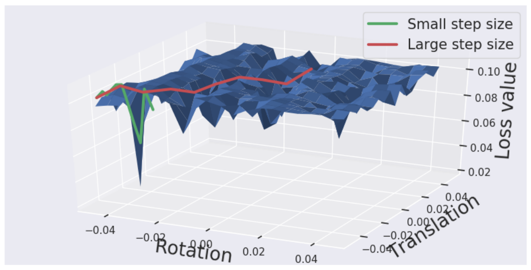


Figure 9: Loss surface over transformation parameters θ (rotation) and τ_x (translation). Using a small step size (green line) will trap the optimizing process near a local extreme while choosing a large step size (red line) will be less effective.

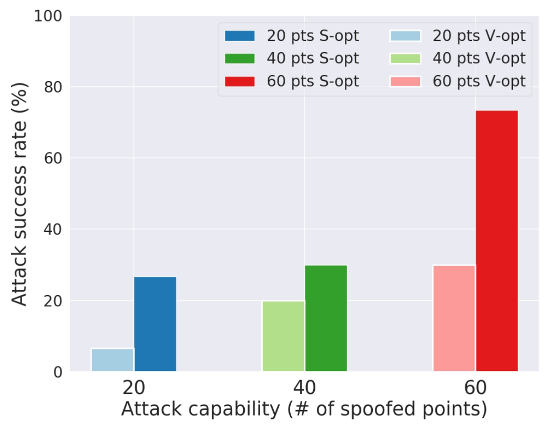


Figure 10: Attack success rate of spoofing a front-near obstacle with different number of spoofed points. V-opt refers to vanilla optimization which is directly using the optimizer and S-opt refers to sampling based optimization. We choose Adam [35] as the optimizer in both cases.

8 EVALUATION AND RESULTS

In this section, we evaluate our adversarial example generation method in terms of attack effectiveness and robustness.

Experiment Setup. We use the real-world LiDAR sensor data trace released by Baidu Apollo team with Velodyne HDL-64E S3, which is collected for 30 seconds on local roads at Sunnyvale, CA. We uniformly sample 300 3D point cloud frames from this trace in our evaluation. The attack goal is set as spoofing an obstacle that is 2-8 meters to the front of the victim AV. The distance is measured from the front end of the victim AV to the rear end of the obstacle.

8.1 Attack Effectiveness

Fig. 10 shows the success rates of generating a spoofed obstacle with different attack capabilities using the vanilla optimization and our improved optimization with global sampling (detailed in §7). As shown, with our improvement using sampling, the success rates of spoofing front-near obstacles are increased from 18.9% to 43.3% on average, which is a 2.65× improvement. This shows that combining global sampling with optimization is effective in addressing the problem of trapping in local minima described in §7.

Fig. 10 also shows that the success rates increase with more spoofed points, which is expected since the attack capability is increased with more spoofed points. In particular, when the attacker can reliably inject 60 spoofed points, which is the attack capability observed in our experiments (§4), the attack success rate is able to achieve around 75% using our improved optimization method.

In addition, we observe that the spoofed obstacles in all of the successful attacks are classified as vehicles after the LiDAR-based perception process, even though we do not specifically aim at spoofing vehicle-type obstacles in our problem formulation.

8.2 Robustness Analysis

In this section, we perform analysis to understand the robustness of the generated adversarial spoofed 3D point cloud T' to variations in 3D point cloud X and spoofed 3D point cloud $T \in \mathcal{S}_T$. Such analysis is meaningful for generating adversarial spoofed 3D point cloud that has high attack success rate in the real world. To launch the

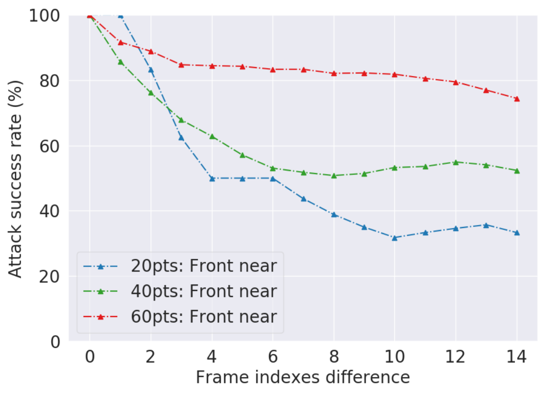


Figure 11: The robustness of the generated adversarial spoofed 3D point cloud to variations in 3D point cloud X. We quantify the variation in 3D point cloud X as the frame indexes difference between the evaluated 3D point cloud and the 3D point cloud used for generating the adversarial spoofed 3D point cloud.

attack in the real world, there are two main variations that affect the results: variation in spoofed points and variation in positions of the victim AV. 1) The imprecision in the attack devices contributes to the variation of the spoofed points. The attacker is able to stably spoof 60 points at a global position as we state in §2.2. However, it is difficult to spoof points with precise positions. It is important to understand whether such imprecision affects the attack success rate. 2) The position of the victim AV is not controlled by the attacker and might vary from where the attacker collected the 3D point cloud. It is important to understand whether such difference affects the attack success rate.

Robustness to variations in point cloud. To measure the robustness to variations in the 3D point cloud, we first select all the 3D point cloud frames that can generate successful adversarial spoofed 3D point cloud. For each of them, we apply its generated adversarial spoofed 3D point cloud to 15 consecutive frames (around 1.5 s) after it and calculate the success rates. Fig. 11 shows the analysis results. In this figure, the x-axis is the index for the 15 consecutive frames, and thus the larger the frame index is, the larger the variation is to the original 3D point cloud that generates the adversarial spoofed 3D point cloud. As shown, the robustness for attacks with more spoofed points is generally higher than that for attacks with fewer spoofed points, which shows that higher attack capability can increase the robustness. Particularly, with 60 spoofed points, the success rates are on average above 75% during the 15 subsequent frames, which demonstrates a high degree of robustness. This suggests that launching such attack does not necessarily require the victim AV to appear at the exact position that generates the adversarial example in order to have high success rates.

Robustness to variations in spoofed 3D point cloud. To evaluate the robustness to variations in the spoofed 3D point cloud, for a given spoofed 3D point cloud $T \in \mathcal{S}_{\mathrm{T}}$, we first generate the corresponding adversarial spoofed 3D point cloud T' with a 3D point cloud X. Next, we generate 5 more spoofed 3D point cloud

Targeted position	# Spoofed points		
	20	40	60
2-8 meters	87%	82%	90%

Table 4: Robustness analysis results of generated adversarial spoofed 3D point cloud to variation in spoofed 3D point cloud $T \in \mathcal{S}_{\mathrm{T}}$. The robustness is measured by average attack success rates.

traces $T_1, \ldots, T_5$ using our LiDAR spoofing attack experiment setup. Next, we use the same transformation that generates T' from T to generate $T'_1, \ldots, T'_5$, and then combine each of them with X to launch the attack. Table 4 shows the average success rates with different attack capabilities. As shown, for all three attack capabilities we are able to achieve over 82% success rates. With 60 spoofed points, the success rate is as high as 90%. This suggests that launching such attack does not require the LiDAR spoofing attack to be precise all the time in order to achieve high success rates.

9 DRIVING DECISION CASE STUDY

To understand the impact of our attacks at the driving decision level, in this section we construct several attack scenarios and evaluate them on Baidu Apollo using simulation as case studies.

Experiment setup. We perform the case study using the simulation feature provided by Baidu Apollo, called *Sim-control*, which is designed to allow users to observe the AV system behavior at the driving decision level by replaying collected real-world sensor data traces. Sim-control does not consist of a physics engine to simulate the control of the vehicle. Instead, the AV behaves exactly the same as what it plans. Although it cannot directly reflect the attack consequences in the physical world, it can serve for our purpose of understanding the impact of our attacks on AV driving decisions.

For each attack scenario in the case study, we simulate it in Sim-control using synthesized continuous frames of successful adversarial 3D point cloud identified in § 8 as input. The experiments are performed on Baidu Apollo 3.0.

Case study results. We construct and evaluate two attack scenarios in this case study[1]:

(1) Emergency brake attack. In this attack, we generate adversarial 3D point cloud that spoofs a front-near obstacle to a moving victim AV. We find that the AV makes a stop decision upon this attack. As illustrated in Fig. 12, the stop decision triggered by a spoofed front-near obstacle causes the victim AV to decrease its speed from 43 km/h to 0 km/h within 1 second. This stop decision will lead to a hard brake [1], which may hurt the passengers or result in rear-end collisions. Noticed that, Apollo does implement driving decisions for overtaking. However, for overtaking, a minimum distance is required based on the relative speed of the obstacle. Therefore, with our near spoofed obstacle, the victim AV makes stop decisions instead of overtaking decisions.

(2) AV freezing attack. In this attack, we generate an adversarial 3D point cloud that spoofs an obstacle in front of an AV victim when it is waiting for the red traffic light. We simulate this scenario with the data trace at an intersection with traffic lights. As shown in Fig. 13, since the victim AV is static, the attacker can constantly

[1]Video demos can be found at http://tinyurl.com/advlidar

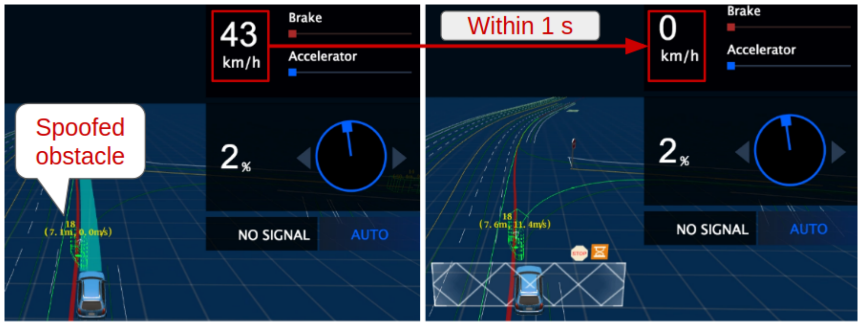


Figure 12: Demonstration of the emergency brake attack. Due to the spoofed obstacle, the victim AV makes a sudden stop decision to drop its speed from 43 km/h to 0 km/h within a second, which may cause injuries of passengers or rear-end collisions.

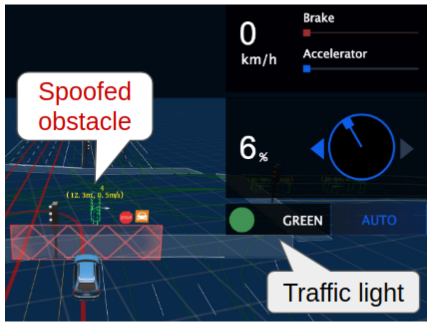


Figure 13: Demonstration of the AV freezing attack. The traffic light is turned green but the victim AV is not moving due to the spoofed front-near obstacles.

attack and prevent it from moving even after the traffic signal turns green, which may be exploited to cause traffic jams. Noticed that, Apollo does implement driving decisions for deviating static obstacles. However, for deviation or side passing, it requires a minimum distance (15 meters by default). Therefore, with our near spoofed obstacle, the victim AV makes stop decisions instead of side passing decisions.

10 DISCUSSION

In this section, we discuss the limitations and generality of this study. We then discuss potential defense directions.

10.1 Limitations and Future Work

Limitations in the sensor attack. One major limitation is that our current results cannot directly demonstrate attack performance and practicality in the real world. For example, performing our attack on a real AV on the road requires dynamically aiming an attack device at the LiDAR on a victim car with high precision, which is difficult to prove the feasibility without road tests in the physical world. In this work, our goal is to provide new understandings of this research problem. Future research directions include conducting real world testing. To demonstrate the attack in the real world, we plan to first conduct the sensor attack with LiDAR on top of a real vehicle in outdoor settings. In this setting, the sensor attack could be enhanced by: 1) enlarging the laser spoofing area to solve the aiming problem; 2) adjusting the delay time so that the attacker could spoof points at different angles without moving the attack devices. Then we could apply our proposed methodology to conduct drive-by experiments in different attack scenarios mentioned in §9.

Limitations in adversarial example generation. First, we construct adversarial sensor data by using a subset of spoofing attack capability $\mathcal{A}$. Therefore, our analysis may not fully reveal the full potential of sensor attacks. Second, though we have performed the driving decision case study, we did not perform a comprehensive analysis on modules beyond the perception module. That means that the designed objective function can be further improved to more directly target specific abnormal AV driving decisions.

10.2 Generality on LiDAR-based AV Perception

Generality of the methodology. Attacking any LiDAR-based AV perception system with an adversarial sensor attack can be formulated as three components: (1) formulating the spoofed 3D point cloud capability $\mathcal{A}$, (2) generating adversarial examples, and (3) evaluating at the driving decision level. Even though our construction of these components might be specific to Baidu Apollo, our analysis methodology can be generalized to other LiDAR-based AV perception systems.

Generality of the results. The formulation of 3D point cloud spoofing capability $\mathcal{A}$ can be generalized as it is independent from AV systems. The success of the attack may be extended to other LiDAR-based AV perception system due to the nature of the LiDAR sensor attack. The LiDAR spoofing attack introduces a spoofed 3D point cloud, which was not foreseen in the training process of machine learning models used in the AV perception system. Therefore, other models are likely to be also vulnerable to such spoofing patterns.

10.3 Defense Discussion

This section discusses defense directions at AV system, sensor, and machine learning model levels.

10.3.1 AV System-Level defenses.

In our proposed attack, the attacker only needs to inject at most 60 points to spoof an obstacle, but the 3D point cloud of a detected real vehicle can have as many as a thousand points (can be illustrated in Fig. 6). We look into the point cloud of a detected spoofed obstacle and find that the 3D point cloud consists of points reflected from the ground, in addition to the points spoofed by the attacker. For example, one of the successful adversarial spoofed 3D point cloud we generated with 20 spoofed points is detected as an obstacle containing 283 points.

Points from ground reflection are clustered into obstacles due to the information loss introduced in the pre-processing phase. More specifically, mapping a 3D point cloud into a 2D matrix results in height information loss. This vulnerability contributes to the success of the proposed attack. To mitigate the impacts of this problem, we propose two defenses at the AV system level: (1) filtering out the ground reflection in the pre-processing phase, and (2) either avoiding transforming 3D point cloud into input feature matrix or adding more features to reduce the information loss.

10.3.2 Sensor-Level Defenses. Several defenses could be adopted against spoofing attacks on LiDAR sensors:

Detection techniques. Sensor fusion, which intelligently combines data from several sensors to detect anomalies and improve performance, could be adopted against LiDAR spoofing attacks. AV systems are often equipped with sensors beyond LiDAR. Camera, radars, ultrasonic sensors provide additional information and redundancy to detect and handle an attack on LiDAR.

Different sensor fusion algorithms have been proposed focusing on the security and safety aspects [56] [33]. However, the sensor fusion defense requires the majority of sensors to be functioning correctly. While not a perfect defense, sensor fusion approaches can significantly increase the effort of an attacker.

Mitigation techniques. Another class of defenses aims to reduce the influence of the attack by modifying the internal sensing structure of the LiDAR. Different solutions include reducing the receiving angle and filtering unwanted light spectra to make LiDARs less susceptible to attacks [42, 44]. However, these techniques also reduce the capacity of the LiDAR to measure the reflected laser pulses, which limits the range and the sensitivity of the device.

Randomization techniques. Another defense is adding randomness to how the LiDAR fires laser pulses. The attacker cannot know when to and what laser pulses to fire if the LiDAR fires laser pulses with an unpredictable pattern. A solution could be firing a random grouping of laser pulses each cycle. An attacker would not know which reflections the LiDAR would be expecting. Another alternative would be randomizing the laser pulses waveform. With sensitive equipment, it would be possible to only accept reflection waveforms that match randomized patterns uniquely produced by the LiDAR laser. Another solution, proposed by Shoukry et al. [46], consists of randomly turning off the transmitter to verify with the receiver if there are any unexpected incoming signals. Adding randomness makes it difficult for an attacker to influence the measurements, but this approach also adds significant complexity to the overall system and trades off with performance.

10.3.3 Machine Learning Model-Level Defense. Various detection and defense methods have also been explored [16, 19, 38, 39] against adversarial examples in image classification. Adversarial training [31] and its variations [39, 47] are more successful to improve the robustness of the model. Motivated by the adversarial examples generated by our algorithm, we can combine them with the original training data to conduct adversarial retraining and thus improve the model robustness.

11 RELATED WORK

Vehicle systems security. Numerous previous works explore security problems in vehicle systems and have uncovered vulnerabilities in in-vehicle networks of modern automobiles [22, 25, 36], infotainment systems [40], and emerging connected vehicle-based systems [23, 30, 48]. In comparison, our work focuses on vehicle systems with the emerging autonomous driving technology and specifically targets the security of LiDAR-based AV perception, which is an attack surface not presented in traditional vehicle systems designed for human drivers.

Vehicle-related sensor attacks. The sensors commonly used in traditional vehicles have been shown to be vulnerable to attacks. Rouf et al. showed that tire pressure sensors are vulnerable to wireless jamming and spoofing attacks [43]. Shoukry et al. attacked the anti-lock braking system of a vehicle by spoofing the magnetic wheel speed sensor [45]. As AVs become popular, so have attacks against their perception sensors. Yan et al. used spoofing and jamming attacks to attack the ultrasonic sensors, radar, and camera on a Tesla Model S [55]. There have also been two works exploring the vulnerability of LiDAR to spoofing and jamming attacks [42, 44]. In this work, we build on these prior work to show that LiDAR spoofing attacks can be used to attack the machine learning models used for LiDAR-based AV perception and affect the driving decision.

Adversarial example generation. Adversarial examples have been heavily explored in the image domain [21, 31, 41, 52]. Xie et al. [53] generated adversarial examples for segmentation and object detection while Cisse et al. [26] for segmentation and human pose estimation. Researchers also apply adversarial examples to the physical world to fool machine learning models [17, 27, 28]. Compared to these previous work exploring adversarial examples in the image domain, this work explores adversarial examples for LiDAR-based perception. An ongoing work [49] studies the generation of 3D adversarial point clouds. However, such attack focuses on the digital domain and can not be directly applied to the context of AV systems. In comparison, our method is motivated to generate adversarial examples based on the capability of sensor attacks to fool the LiDAR-based perception models in AV systems.

12 CONCLUSION

In this work, we perform the first security study of LiDAR-based perception in AV systems. We consider LiDAR spoofing attacks as the threat model, and set the attack goal as spoofing front-near obstacles. We first reproduce the state-of-the-art LiDAR spoofing attack, and find that blindly applying it is insufficient to achieve the attack goal due to the machine learning-based object detection process. We thus perform analysis to fool the machine learning model by formulating the attack task as an optimization problem. We first construct the input perturbation function using local attack experiments and global spatial transformation-based modeling, and then construct the objective function by studying the post-processing process. We also identify the inherent limitations of directly using optimization-based methods and design a new algorithm that increases the attack success rates by 2.65× on average. As a case study, we further construct and evaluate two attack scenarios that may compromise AV safety and mobility. We also discuss defense directions at AV system, sensor, and machine learning model levels.

ACKNOWLEDGMENTS

We would like to thank Shengtuo Hu, Jiwon Joung, Yunhan Jack Jia, Yuru Shao, Yikai Lin, David Ke Hong, the anonymous reviewers, and our shepherd Zhe Zhou for providing valuable feedback on our work. This research was supported in part by an award from Mcity at University of Michigan, by the National Science Foundation under grants CNS-1850533, CNS-1330142, CNS-1526455 and CCF-1628991, by ONR under N00014-18-1-2020.

REFERENCES

[1] 2005. HARD BRAKE HARD ACCELERATION. http://tracknet.accountsupport.com/wp-content/uploads/Verizon/Hard-Brake-Hard-Acceleration.pdf. (2005).

[2] 2016. ArbExpress. https://www.tek.com/signal-generator/afg2021-software-0. (2016).

[3] 2017. An Introduction to LIDAR: The Key Self-Driving Car Sensor. https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff. (2017).

[4] 2017. Baidu Apollo. http://apollo.auto. (2017).

[5] 2017. Google's Waymo Invests in LIDAR Technology, Cuts Costs by 90 Percent. https://arstechnica.com/cars/2017/01/googles-waymo-invests-in-lidar-technology-cuts-costs-by-90-percent/. (2017).

[6] 2017. KITTI Vision Benchmark: 3D Object Detection. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. (2017).

[7] 2017. What it Was Like to Ride in GM's New Self-Driving Cruise Car. https://www.recode.net/2017/11/29/16712572/general-motors-gm-new-self-driving-autonomous-cruise-car-future. (2017).

[8] 2018. Baidu hits the gas on autonomous vehicles with Volvo and Ford deals. https://techcrunch.com/2018/11/01/baidu-volvo-ford-autonomous-driving/. (2018).

[9] 2018. Baidu starts mass production of autonomous buses. https://www.dw.com/en/baidu-starts-mass-production-of-autonomous-buses/a-44525629. (2018).

[10] 2018. VeloView. https://www.paraview.org/VeloView/. (2018).

[11] 2018. Volvo Finds the LIDAR it Needs to Build Self-Driving Cars. https://www.wired.com/story/volvo-self-driving-lidar-luminar/. (2018).

[12] 2018. Waymo's autonomous cars have driven 8 million miles on public roads. https://www.theverge.com/2018/7/20/17595968/waymo-self-driving-cars-8-million-miles-testing. (2018).

[13] 2018. What Is LIDAR, Why Do Self-Driving Cars Need It, And Can It See Nerf Bullets? https://www.wired.com/story/lidar-self-driving-cars-luminar-video/. (2018).

[14] 2018. You can take a ride in a self-driving Lyft during CES. https://www.theverge.com/2018/1/2/16841090/lyft-aptiv-self-driving-car-ces-2018. (2018).

[15] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

[16] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).

[17] Anish Athalye and Ilya Sutskever. 2018. Synthesizing Robust Adversarial Examples. In *International Conference on Machine Learning (ICML)*.

[18] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *USENIX Security Symposium*.

[19] Nicholas Carlini and David Wagner. 2017. Adversarial Examples are not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 3–14.

[20] Nicholas Carlini and David Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-text. In *Deep Learning and Security Workshop (DLS)*.

[21] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. 39–57. https://doi.org/10.1109/SP.2017.49

[22] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*.

[23] Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Z. Morley Mao, and Henry X. Liu Liu. 2018. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS '18)*.

[24] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples. *arXiv preprint arXiv:1803.01128* (2018).

[25] Kyong-Tak Cho and Kang G. Shin. 2016. Error Handling of In-vehicle Networks Makes Them Vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*.

[26] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373* (2017).

[27] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* 1 (2017).

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

[29] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Yiheng Feng, Shihong Huang, Qi Alfred Chen, Henry X. Liu, and Z. Morley Mao. 2018. Vulnerability of Traffic Control System Under Cyber-Attacks Using Falsified Data. In *Transportation Research Board 2018 Annual Meeting (TRB)*.

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

[32] Velodyne LiDAR Inc. 2018. VLP-16 User Manual. (2018).

[33] Radoslav Ivanov, Miroslav Pajic, and Insup Lee. 2014. Attack-resilient sensor fusion. In *Proceedings of the conference on Design, Automation & Test in Europe*.

[34] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. In *Advances in neural information processing systems*. 2017–2025.

[35] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[36] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. 2010. Experimental Security Analysis of a Modern Automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP'10)*.

[37] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Annual Network and Distributed System Security Symposium (NDSS)*.

[38] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. 2018. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. *arXiv preprint arXiv:1801.02613* (2018).

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[40] Sahar Mazloom, Mohammad Rezaeirad, Aaron Hunter, and Damon McCoy. 2016. A Security Analysis of an In-Vehicle Infotainment and App Platform. In *Usenix Workshop on Offensive Technologies (WOOT)*.

[41] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks Against Machine Learning. In *ACM on Asia Conference on Computer and Communications Security*.

[42] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. 2015. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. In *Black Hat Europe*.

[43] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. 2010. Security and Privacy Vulnerabilities of In-car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *Proceedings of the 19th USENIX Conference on Security (USENIX Security'10)*. USENIX Association, Berkeley, CA, USA, 21–21. http://dl.acm.org/citation.cfm?id=1929820.1929848

[44] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. 2017. Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*.

[45] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Non-invasive Spoofing Attacks for Anti-lock Braking Systems. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, Guido Bertoni and Jean-Sébastien Coron (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 55–72.

[46] Yasser Shoukry, Paul Martin, Yair Yona, Suhas N. Diggavi, and Mani B. Srivastava. 2015. PyCRA: Physical Challenge-Response Authentication For Active Sensors Under Spoofing Attacks. In *ACM Conference on Computer and Communications Security*.

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

[48] Wai Wong, Shihong Huang, Yiheng Feng, Qi Alfred Chen, Z Morley Mao, and Henry X Liu. 2019. Trajectory-Based Hierarchical Defense Model to Detect Cyber-Attacks on Transportation Infrastructure. In *Transportation Research Board 2019 Annual Meeting (TRB)*.

[49] Chong Xiang, Charles R Qi, and Bo Li. 2018. Generating 3D Adversarial Point Clouds. *arXiv preprint arXiv:1809.07016* (2018).

[50] Chaowei Xiao, Ruizhi Deng, Bo Li, Fisher Yu, Dawn Song, et al. 2018. Characterizing Adversarial Examples Based on Spatial Consistency Information for Semantic Segmentation. In *Proceedings of the (ECCV)*. 217–234.

[51] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610* (2018).

[52] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. 2018. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*

(2018).
[53] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. 2017. Adversarial Examples for Semantic Segmentation and Object Detection. In *IEEE International Conference on Computer Vision (ICCV)*.
[54] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darell, and Dawn Song. 2017. Can you fool AI with adversarial examples on a visual Turing test? *arXiv preprint arXiv:1709.08693* (2017).
[55] Chen Yan. 2016. Can You Trust Autonomous Vehicles : Contactless Attacks against Sensors of Self-driving Vehicle.
[56] Kang Yang, Rui Wang, Yu Jiang, Houbing Song, Chenxia Luo, Yong Guan, Xiaojuan Li, and Zhiping Shi. 2018. Sensor attack detection using history based pairwise inconsistency. *Future Generation Computer Systems* (2018).
[57] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. In *USENIX Security Symposium*.

APPENDIX

A ALGORITHM DETAILS AND EXPERIMENT SETTINGS

Algorithm 1 shows the detailed algorithm to generate adversarial examples. In our experiment, we select Adam [35] as our optimizer *opt* with learning rate $1e-4$. $opt(l_{\text{adv}}; \theta, \tau_x, s_h)$ means updating the parameters $\theta, \tau_x, s_h)$ with respect to Loss function l_{adv}. We select TensorFlow [15] as backbone. L_t is set as 12.5 while L_θ is set as the angle that generates 2-meter distance from the target position.

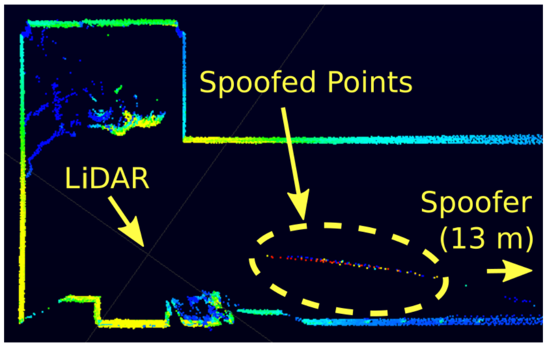


Figure 14: Collected traces from the reproduced sensor attack. The points in the yellow circle are spoofed by the sensor attack.

Algorithm 1: Generating adversarial examples by leveraging global spatial transformation

```
   input: Target model: $M$;
          3D point cloud $X$ ;
1         3D spoofed 3D point cloud $T$;
          Optimizer $opt$;
          Max iteration $N$;
   output: 3D adversarial 3D point cloud $X'$;
2  Initialization: $\theta \leftarrow 0, \tau_x \leftarrow 0, s_h \leftarrow 1, l_{min} = +\text{inf}$,
     $x = \Phi(X), t = \Phi(T)$;
   /* Initiate parameters by sampling around the
      transformation parameters $Target_\theta$, $Target_{\tau_x}$ that
      transforms t to the target position $(px, py)$ of
      the attack                                          */
3  for $i\tau_x \leftarrow -L_t$ to $L_t$ do
4      for $i\theta \leftarrow -L_\theta$ to $L_\theta$ do
           /* Initialize parameter .          */;
5          $\theta \leftarrow Target_\theta + i\theta, \tau_x \leftarrow Target_{\tau_x} + \tau_x i$;
6          for $iter \leftarrow 1$ to $N$ do
               /* Calculate adversarial loss      */;
7              $l_{\text{adv}} \leftarrow$ Equation 7.;
               /* Update the parameters $\theta, \tau_x, s_h$ based on
                  optimizer $opt$ and loss $l_{\text{adv}}$          */
8              ;
9              $\theta, \tau_x, s_h \leftarrow opt(l_{\text{adv}}; \theta, \tau_x, s_h)$
10             if $l_{min} < l_{adv}$ then
11                 $\theta^{final}, \tau_x^{final}, s_h^{final} \leftarrow \theta, \tau_x, s_h$
12         end
13     end
14 end
15 $T' \leftarrow G_T(\theta^{final}, t_x^{final}, s_h^{final}; T)$;
16 $X' \leftarrow X + T'$;
   Return: $T'$
```