Generating 3D Adversarial Point Clouds under the Principle of LiDARs

Bo Yang[1], Yushi Cheng[2], Zizhi Jin[1], Xiaoyu Ji[1†], Wenyuan Xu[1]
[1] Ubiquitous System Security Lab (USSLAB), Zhejiang University
[2] Research Center for Information Science and Technology (BNRist), Tsinghua University
[1]{yb5, zizhi, xji, wyxu}@zju.edu.cn, [2] yushithu@mail.tsinghua.edu.cn

***Abstract*—Due to the booming of autonomous driving, in which LiDAR plays a critical role in the task of environment perception, its reliability issues have drawn much attention recently. LiDARs usually utilize deep neural models for 3D point cloud perception, which have been demonstrated to be vulnerable to imperceptible adversarial examples. However, prior work usually manipulates point clouds in the digital world without considering the physical working principle of the actual LiDAR. As a result, the generated adversarial point clouds may be realizable and effective in simulation but cannot be perceived by physical LiDARs. In this work, we introduce the physical principle of LiDARs and propose a new method for generating 3D adversarial point clouds in accord with it that can achieve two types of spoofing attacks: object hiding and object creating. We also evaluate the effectiveness of the proposed method with two 3D object detectors on the KITTI vision benchmark.**

I. Introduction

With the booming of autonomous driving, major companies have launched their self-driving products [1]–[6], which rely on a variety of sophisticated sensors such as cameras, radars, LiDARs, and ultrasonic sensors [7] to perceive the surrounding complex road information. Among those sensors, LiDARs provide richer environmental depth information and enable 3D perception with the help of 3D object detectors, and thus are widely deployed on various autonomous vehicles. As a result, to ensure the safety of autonomous driving decisions, it is critical for 3D object detectors to analyze the point clouds collected by LiDARs reliably.

LiDAR-based 3D object detectors are usually based on Deep Neural Networks (DNNs), which, however, have been demonstrated to be vulnerable to adversarial examples [9], [10], [13], [16], [19]. Adversaries can induce DNN models into wrong decisions by adding tiny, imperceptible disturbances to images, voices, texts, etc. LiDAR-based 3D object detectors have also been proved to suffer from such attacks [8], [20]. However, deceiving LiDAR-based 3D detectors in the physical world requires not only the capability of exploiting the vulnerability of DNN models to generate adversarial point clouds but also the ability to ensure the generated adversarial point clouds are perceived by the actual LiDARs. Most existing methods focus on developing the first capability and manipulating point clouds by adding [8], disturbing [21], or moving points [14], [18] in the digital world without considering the physical working principle of the actual LiDARs. As a result, the generated point clouds may be effective in simulation but cannot be perceived by physical LiDARs. Several other works have attempted to physically realize adversarial point clouds optimized in the digital world by placing a 3D printed physical object [20] or a drone [23] near the target object. However, such attacks may raise the suspicion of alert users.

[†]Corresponding author

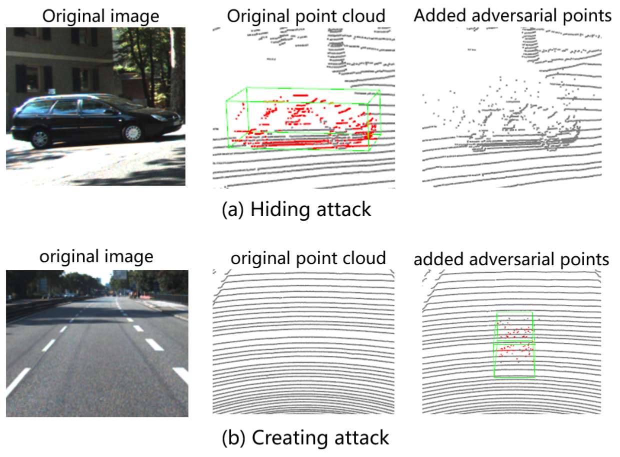

Fig. 1. Examples of hiding attack (a) and creating attack (b). By adding adversarial points that comply with the LiDAR principle above the target object, we successfully make the 3D target detector unable to recognize the original object. By adding adversarial points that comply with the LiDAR principle on the open road, we successfully fool the 3D detector to create a target.

In this paper, we try to answer such a question: *How to ensure that the generated adversarial points can be perceived by LiDARs in the physical world?* One thing that needs attention is that the actual point cloud space of a LiDAR is sparse. A mechanical (spinning) LiDAR usually has multiple transmitters and receivers at the vertical angle and can obtain sparse 64-line point (or 16-line, 32-line, etc.) cloud data by emitting laser rays and receiving reflected light. LiDARs deployed in autonomous vehicles operate under the Strongest Return Mode setting [8], [17], [18], in which each laser ray can

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022
24 April 2022, San Diego, CA, USA
ISBN 1-891562-75-4
https://dx.doi.org/10.14722/autosec.2022.23026
www.ndss-symposium.org

record at most one point. Therefore, to ensure that LiDARs can perceive adversarial points emitted by the attack transmitter in the physical world, we shall limit the locations of the generated adversarial points to specific LiDAR rays.

Based on this physical constraint, we propose a new method to generate adversarial point clouds that can be injected into the victim LiDAR by a laser transmitter. Our attack scenario is as follows: An adversary tries to inject malicious points into the victim LiDAR through the laser transmitter on the roadside to induce hiding attacks (HA) and creating attacks (CA) that affect safety-critical decision making. As shown in Fig.1, the goal of HA is to make the detector fail to perceive the existing object, which may induce the victim autonomous vehicle into a collision. To achieve it, we add adversarial points in the space above the object and update the distance of the adversarial points on the laser ray according to the gradient information, which will not block the point cloud of the original target, facilitating attacks in the physical world. For CA, our goal is to render the detector perceive non-existent targets, which can induce the moving victim vehicle into a sudden brake. To achieve it, we randomly add points to the laser ray in a limited space (like the size of the detection frame) and update the distance of the adversarial point on the ray according to the gradient information until the detector misjudges the adversarial point specified by the adversary.

To validate our attack method, we evaluate with two 3D object detectors PointPillars [15] and SECOND [22]. In summary, our contributions include the points below:

- We propose a new LiDAR-principle-orientated optimization method to generate 3D adversarial point clouds that can achieve both object hiding and object creating attack effects.
- We validate the effectiveness of our attack methods with two 3D object detectors PointPillars and SECOND.

II. Background

A. LiDAR-based 3D Object Detection

State-of-the-art 3D object detection models are usually based on deep learning techniques and have three main categories: (1) bird's-eye view (BEV) based methods that take point cloud's BEV representation as model input and use 2D Convolutional Neural Networks (CNNs) in feature learning, (2) voxel-based methods that divide the 3D point cloud space into voxels and learn features through 3D CNNs, and (3) point-wise methods that directly operate on point clouds to learn the features. Among these methods, the voxel-based ones are commonly used, and we study their representative models PointPillars [15] and SECOND [22] in this paper.

B. Adversarial Machine Learning

Recently, much attention has been devoted to adversarial attacks that utilize the vulnerabilities of machine learning algorithms, and researchers have proposed various ways to construct adversarial examples (images) that can cause misclassification in 2D image classification and object detection [10], [13], [19]. With the rapid development of 3D

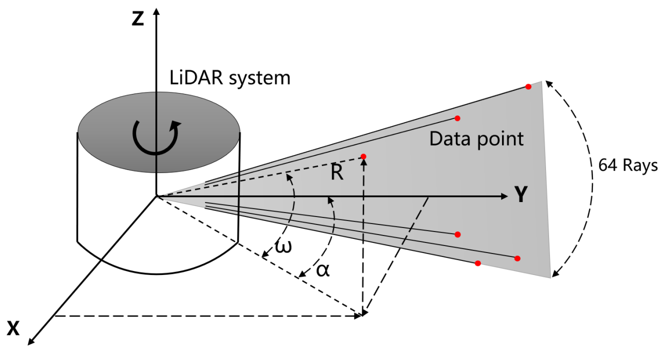

Fig. 2. Schematic diagram of point cloud data acquired by 64-line mechanical LiDAR. The laser rays are sparse and limited by the rotation speed of the LiDAR and the number of internal transmitters. Every generated point only occurs on one of the LiDAR's laser rays, and each laser ray has at most one point.

perception, adversarial machine learning against LiDAR-based 3D object detection begins to draw attention as well [8], [18], [20], [23].

For a general 3D object detector, we can model its adversarial attack as:

$$Y' = f(X + \Delta) \neq Y \tag{1}$$

where f is the abstract function of the 3D object detector, $X \in \mathbb{R}^{n\times 4}$ is the benign input point cloud (n is the number of input points, and each point is represented as its 3D coordinates and light intensity), $Y \in \mathbb{R}^{m\times t}$ is the benign output of the detector (m is the number of objects detected in the point cloud, and each object is detected with a t-dimension vector indicating its predicted location, predicted class, and class prediction confidence), Δ is the additive noise (e.g., the injected laser points in our case) that has the same dimensionality with X, and Y' is the adversarial output. $X+\Delta$ can be further denoted as X', which is called an adversarial example to the object detector, i.e., an input to machine learning models that an adversary has intentionally designed to cause the model to make a mistake.

III. Attack Design

We propose a novel method to generate adversarial point clouds under the principle of LiDARs. This method is a model-level white-box attack that can achieve the effect of both object creating and object hiding. Next, we will elaborate on the threat model and attack methodology.

A. Threat Model

We assume that the adversary can acquire and analyze the parameters of the 3D target detector deployed on the victim autonomous vehicle, the point cloud data collected by the LiDAR in the current road environment, and the detector output results. In other words, this is a white box attack. In addition, the adversary masters the highly advanced sensor technology and can inject less than 100 points into the victim LiDAR through the laser transmitter, and the horizontal angle range of the injected adversarial point cloud is less than 10°.

B. Attack Methodology

To generate adversarial points that can be injected into the victim LiDAR through the laser transmitter, it is essential to address the following challenge:

- How to generate adversarial point clouds in line with LiDARs' working principle to ensure that LiDARs in the physical world can perceive the generated points?

To generate adversarial points that are in line with LiDAR's working principle, we consider the optimization-based point cloud generation, which exploits the vulnerability of the 3D object detectors, and has the potential of hiding or creating a target class at any distance with fewer points.

Problem Modeling. As shown in Fig.2, we first introduce the principle of LiDARs, a physical constraint that shall be considered during the generation to ensure the physical realizability of the generated point cloud: *The laser rays are sparse and limited by the rotation speed of the LiDAR and the number of internal transmitters. Every generated point only occurs on one of the LiDAR's laser rays, and each laser ray has at most one point.*

To better comply with the physical contrast, we choose to generate the point clouds in the Spherical Coordinate and formulate this problem as a gradient-based optimization problem:

$$\begin{aligned} \min_{\mathrm{P}'} \quad & \mathcal{L}(\mathrm{P}') \\ \text{s.t.} \quad & (R_i', \alpha_i', \omega_i') \in \mathrm{Loc}^{exp}, i \in [1,n] \\ & |\alpha_i' - \alpha_j'| + |\omega_i' - \omega_j'| \neq 0, i,j \in [1,n] \\ & \omega_i' \in \mathbb{W} \end{aligned} \tag{2}$$

where $\mathcal{L}(.)$ is the loss function; $\mathrm{P}' = \{(R_i', \alpha_i', \omega_i') | i \in [1,n]\}$ is the adversarial points; n is the number of adversarial points; R_i', α_i' and ω_i' are the distance, horizontal angle and vertical angle of the ith adversarial point relative to the LiDAR, respectively; $\mathrm{Loc}^{exp} = \{x_a, y_a, z_a, w_a, h_a, l_a, yaw_a\}$ represents the target area for adding adversarial points (x_a, y_a, z_a) is the center point of the target area, (w_a, l_a, h_a) are the width, length and height of the target area, and yaw_a is the yaw angle; $\mathbb{W}$ indicates the range of the vertical angle specified by the victim LiDAR ($i.e.$, 64 discrete values in Velodyne HDL-64E Laserscanner).

Loss Function Design. For HA, our goal is to inject adversarial points into the vicinity of the victim object to make it undetectable. To achieve it, we suppress the bounding box proposals related to the victim objects. A proposal that is close to the victim object can be considered as relevant if (1) their intersection over union (IoU) is larger than a threshold ϵ_i, and (2) the class prediction confidence of the proposal is larger than a threshold ϵ_s. Considering the practicality of the real-world attacks, we choose to inject adversarial points above the victim object to suppress those relevant proposals to avoid the possible blocking from the victim object. In this way, the design of the loss function for HA is as follows:

Algorithm 1: Generating adversarial points

```
   Input: clean point cloud : $P$;
          expected location : $\mathrm{Loc}^{exp}$;
          number of adversarial points: $n$ ;
          number of iterations : $n_{iter}$
1  Randomly initialize adversarial points
     $\mathrm{P}' = \{(R_i', \alpha_i', \omega_i') \in \mathrm{Loc}^{exp} | i \in [1,n]\}$;
2  $P_a \leftarrow P + P'$ ;
3  for iter $\in \{1, 2, 3, ..., n_{iter}\}$ do
4      detection result: IoUs, scores;
5      calculate $\delta'$ by loss function 3 or 4 ;
6      update $P' \leftarrow P' + \delta'$ ;
7      $P_a \leftarrow P + P'$ ;
8  end
   Output: adversarial point cloud : $P_a$
```

$$\mathcal{L}_h = \sum_{b,s \in B} -\mathrm{IoU}(b^t, b)\log(1-s) \tag{3}$$

where $\mathrm{B} = \{(x_i, y_i, z_i, w_i, h_i, l_i, yaw_i, s_i) | i \in [1,k]\}$ is the set of all the bounding box proposals; k is the number of related bounding box proposals; b^t is the ground truth of the victim object, and b and s are the relevant bounding box proposal and the confidence, respectively. In our implementation, $\epsilon_i = 0.1$ and $\epsilon_s = 0.1$.

For CA, our goal is to induce a target object into a specific location by injecting adversarial points into this area, e.g., 10 meters in front of the victim LiDAR. To achieve it, we improve the bounding box proposals related to the expected area. Different from HA, we select the top 10 bounding box proposals that have the largest IoUs with the expected area as the relevant proposals. In this way, we design the loss function for CA as follows:

$$\mathcal{L}_c = \sum_{b,s \in \mathrm{B}} -\mathrm{IoU}(b^e, b)\log(s) \tag{4}$$

where $b^e = \{x_e, y_e, z_e, w_e, h_e, l_e, yaw_e\}$ is the expected area; $\mathrm{B} = \{(x_i, y_i, z_i, w_i, h_i, l_i, yaw_i, s_i) | i \in [1,10]\}$ is the set of top 10 bounding box proposals which have the largest IoUs with the expected area.

Optimization Process. With the loss functions, we then design the optimization process for HA or CA shown in Algorithm 1, which mainly has the following steps:

- Step 1: Calculate the Spherical Coordinate range of the adversarial points according to the location where the adversary expects to hide or create an object;
- Step 2: Randomly add a given number of adversarial points in the range mentioned above;
- Step 3: Calculate the gradient of the loss function for HA or CA to the Spherical Coordinate of the adversarial points;
- Step 4: Update R of the adversarial point cloud P'. Note that we do not update α and ω to ensure that the

TABLE I
RESULTS OF ATTACKS.

Models	Attack Success Rate					
	Hiding Attack			Creating Attack		
	Pedestrian	Cyclist	Car	Pedestrian	Cyclist	Car
PointPillars	100%	100%	84%	51%	50%	51%
SECOND	22%	42%	29%	99%	99%	18%

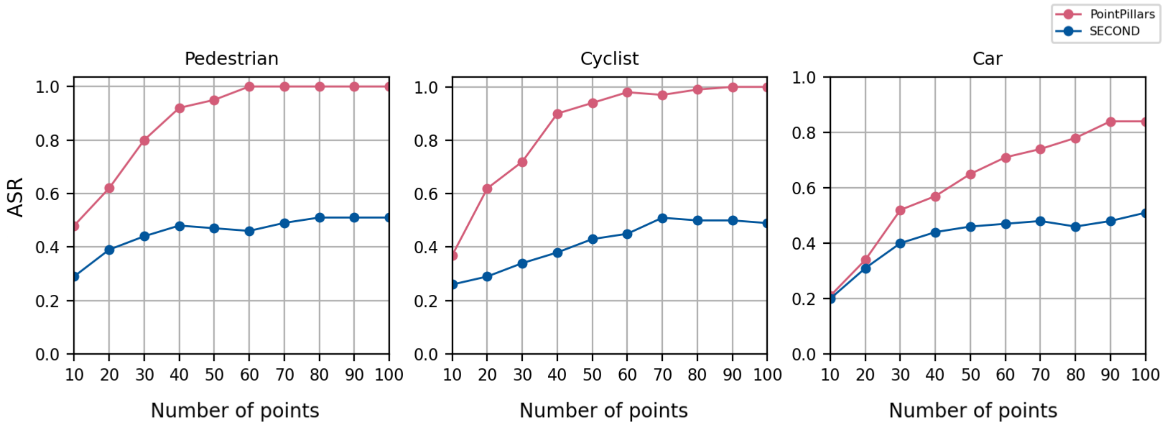

Fig. 3. HA attack success rate (ASR) of 3D object detection for different classes with different numbers of adversarial points.

adversarial points are always on the laser ray, and at most one point appears on a ray.

- Step 5: Repeat Step 3 and Step 4 until the loss converges or the end of the iteration.

The conventional detector processes the point cloud in the Cartesian Coordinate so that we cannot obtain the gradient information about the loss function of the adversarial points in the form of Spherical Coordinate. To overcome it, after adding adversarial points to the clean point cloud, we first convert all the point clouds from the Cartesian Coordinate to Spherical Coordinate, and then into the Cartesian Coordinate.

IV. Evaluation

In this section, we evaluate our attacks against LiDAR-based 3D object detection systems. We consider both simulation evaluation for HA and CA, where the adversarial point clouds fed into the 3D object detectors are generated by optimization under the principle of LiDARs.

We use the attack success rate (ASR) as the metric, the ratio of the number of successful attacks against an object detector over the total number of conducted attacks. We also analyzed the relationship between recall and IoU threshold in the hiding attack. As shown in Table I, we highlight the key result of our attacks as follows:

- For HA, our methods can achieve an overall ASR of 95% against PointPillars and 50% against SECOND.
- For CA, our methods can achieve an overall ASR of 31% against PointPillars and 72% against SECOND.

A. Experimental Setup

Dataset. We use the KITTI [12] dataset in the simulation evaluation, which is widely used in the training and testing of 3D object detectors. For hiding attacks, we select 100 objects for each class of interest from the KITTI dataset and try to make them undetectable. For creating attacks, we select 100 bin files (scenes) from the KITTI dataset and try to inject a car, a pedestrian, or a cyclist into each scene, respectively.

Object Detectors. We evaluate our attacks using two 3D object detectors PointPillars [15] and SECOND [22]. We use the implementation from MMDetection3D [11]. The backbone network and training dataset used for these two pre-trained models are SECFPN and KITTI [12], respectively. The average detection precision achieved on KITTI is 59.5% for PointPillars and 64.4% for SECOND.

Classes of Interest. Given that most LiDAR-based 3D object detectors detect (up to) three classes of objects in the autonomous driving scenarios, which are (1) car, (2) pedestrian, and (3) cyclist, we consider them as classes of interest in this paper. Both the aforementioned object detectors PointPillars [15] and SECOND [22] support the detection of these three classes.

Computing Platform. We implement the aforementioned object detectors in our lab with a server equipped with an Intel Xeon Gold 6240C CPU @2.60 GHz, four GeForce RTX 3090 GPUs, and 256 GB physical memory, which is also used to optimize adversarial point clouds.

B. Attack Effectiveness

In this section, we evaluate the effectiveness of HA and CA, respectively.

Hiding Attacks. The results shown in Fig.3 demonstrate the effectiveness of HA. When against Pointpillars, while the added adversarial points gradually increased from 10 to 60, ASRs of pedestrians and cyclists rise rapidly and stabilize at approximately 100%. For cars, while the number of added adversarial points gradually increased from 10 to 90, the ASRs

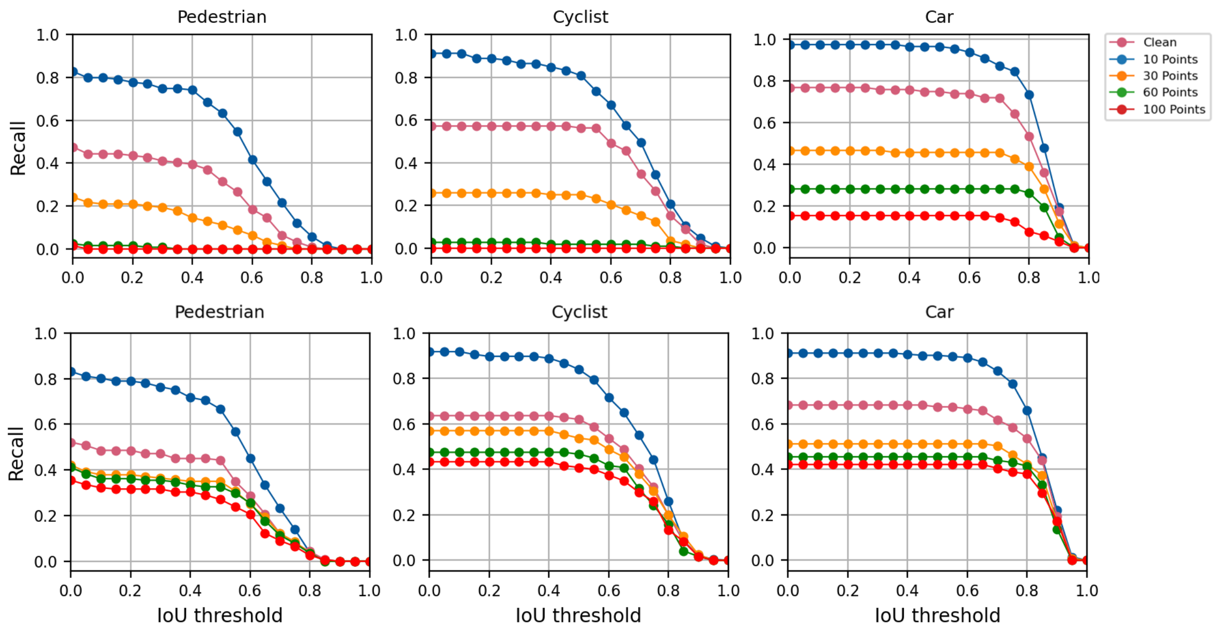

Fig. 4. Recall-IoU curves for (top) PointPillars and (bottom) SECOND under the HA with different IoU thresholds.

rise quickly and then stabilize at 84%. We speculate that more adversarial points are needed to hide the objects such as cars. When against SECOND, the ASRs for pedestrians, cyclists and cars rise slowly with the increase in the number of adversarial points and finally stabilize at 51, 50, and 51, respectively. Generally speaking, the ASRs increase roughly along with the number of spoofing points increasing.

From Fig. 4, we can find that with the increase of the IoU threshold, recall first decreases slowly. When pedestrians IoU threshold > 0.5, cyclist IoU threshold > 0.5 or car IoU threshold > 0.7, recall drops rapidly. As the number of adding adversarial points increases, pedestrians' and cyclists' recalls drop sharply to 0, and the recall of cars drops sharply to 0. As the number of added adversarial points increases, the recall is lower under the same IoU threshold, which confirms that our proposed hiding attack can effectively reduce the performance of the 3D object detector.

Note that the ASRs of HA against the PointPillars model are significantly better than that of the SECOND model. We suppose that the difference comes from these two models' different feature extraction processes. The SECOND model divides the point cloud space into voxels, while PointPillars divides the point cloud space into vertical columns (pillars) and utilizes PointNets to learn features. The location where we add the adversarial points is in the space above the victim object, the adversarial points may be more likely to harm the feature extraction of the point cloud below the pillar.

Creating Attacks. We select the position of the road near the front of (10 meters) the victim LiDAR as the center coordinates of our forged object, and the experimental results are shown in Fig. 5. When against PointPillars, the ASRs for pedestrians, cyclists, and cars can achieve 22%, 45%, and 29%. For these three types of target objects, as the number of added adversarial points increases, the ASRs gradually increase. When against SECOND, the ASRs for pedestrians, cyclists, and cars can achieve 99%, 99%, and 18%. For two types of targets: pedestrians and cyclists, the attack method we propose can easily achieve a nearly 100% ASR within the numbers of added points ranging from 10 to 100. But the attack method we proposed is less effective for targets such as cars. The ASRs of car creating are much lower than that of the other two object types. We guess that the reason may be related to our initial point cloud strategy. When initializing the adversarial points, the target area we choose as the space constraints is a cuboid, which is the same as the bounding box in the object detector, but the point cloud of a real car is distributed on the edge of the cuboid due to the regular shape of a vehicle, while the point clouds of pedestrians and cyclists are more evenly distributed in the detection frame. As a result, we assume that the overhead of creating a car is higher than that of creating a pedestrian or a cyclist by our optimization algorithm.

C. Discussion and Implications

For HA, the performance of our proposed attack method on PointPillars is significantly better than SECOND. We speculate that this is due to the significant difference between the two extracting point cloud data features. How different feature extraction methods will affect the robustness of the detector, and we will continue to dig deeper on this issue. For CA, although our proposed attack method has good performance on two types of targets, pedestrians, and cyclists, it performs poorly on targets such as cars. We guess this is related to the initialization of the added adversarial points. Clean car point clouds are often distributed on the surface of the detection frame. Our attack method randomly distributes the point clouds within the expected detection frame when

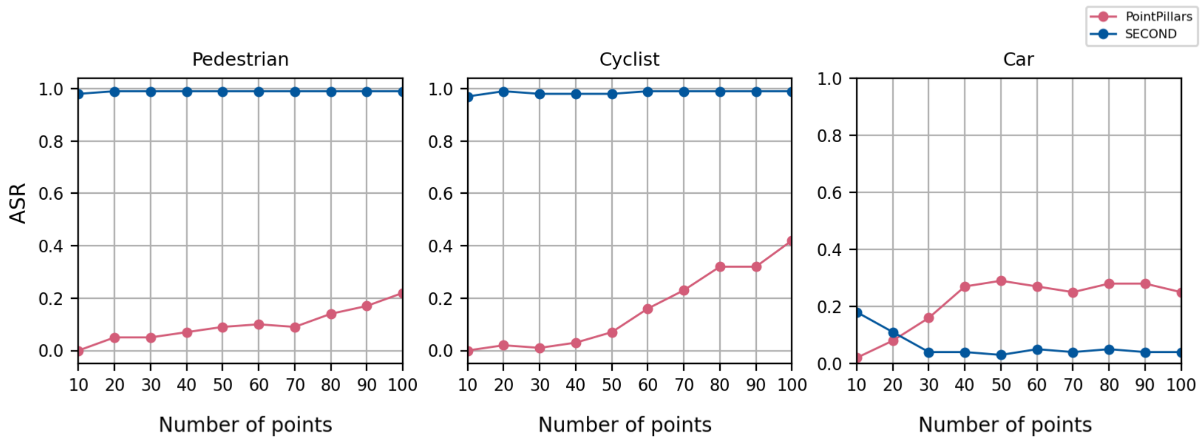

Fig. 5. CA attack success rate (ASR) of 3D object detection for different classes with different numbers of adversarial points.

initializing the adversarial points, making it more challenging to find the optimal solution. While the other two types of point clouds, clean pedestrians and cyclists, are relatively evenly distributed within the detection frame. We hope to stimulate more thinking, such as exploring how to improve the initialization strategy of the algorithm.

V. Conclusion

We consider the principle of LiDAR data collection and propose a novel attack method to generate the 3D point cloud adversarial examples that LiDARs can perceive. Through experiments on two models and three types of objects: pedestrians, cyclists, and cars, it is demonstrated that the attack method we propose can achieve two kinds of attack effects: object hiding and object creating. We also consider the impact of the number of injected adversarial points and find that whether it is an object hiding attack or a creating creation attack, as the number of injection adversarial points increases, the performance of the attack generally increases. In the future, we will explore how to accurately inject the adversarial points obtained at the software level into the victim LiDAR in the physical world to deceive the 3D detector, and continue to explore the safety of autonomous driving based on LiDARs.

Acknowledgments

We thank Yubin Liu for the help in the early stage of this work. This work was supported by China NSFC Grant nos. 62071428, 61941120, 61925109, ZJNSF Grant no. LGG19F020020, and CPSF Grant no. BX2021158.

References

[1] http://www.apollo.auto/.
[2] https://waymo.com/.
[3] https://www.arcfox.com.cn/s-huawei-hi.html.
[4] https://www.nio.cn/nad.
[5] https://www.xiaopeng.com/p5.html.
[6] https://www.mobileye.com/.
[7] https://en.wikipedia.org/wiki/Self-driving_car.
[8] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.
[9] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 513–530.
[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.
[11] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," https://github.com/open-mmlab/mmdetection3d, 2020.
[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.
[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[14] Z. Hau, K. T. Co, S. Demetriou, and E. C. Lupu, "Object removal attacks on lidar-based 3d object detectors," *arXiv preprint arXiv:2102.03722*, 2021.
[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.
[16] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.
[17] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 445–467.
[18] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 877–894.
[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[20] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun, "Physically realizable adversarial examples for lidar object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 716–13 725.
[21] X. Wang, M. Cai, F. Sohel, N. Sang, and Z. Chang, "Adversarial point cloud perturbations against 3d object detection in autonomous driving systems," *Neurocomputing*, vol. 466, pp. 27–36, 2021.
[22] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
[23] Y. Zhu, C. Miao, T. Zheng, F. Hajiaghajani, L. Su, and C. Qiao, "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*, 2021, pp. 1945–1960.