Fooling LiDAR Perception via Adversarial Trajectory Perturbation

Yiming Li[†,*] Congcong Wen[†,§,*] Felix Juefei-Xu[‡] Chen Feng[†]
[†]New York University [§]University of Chinese Academy of Sciences [‡]Alibaba Group
yimingli@nyu.edu, cfeng@nyu.edu

Abstract

LiDAR point clouds collected from a moving vehicle are functions of its trajectories, because the sensor motion needs to be compensated to avoid distortions. When autonomous vehicles are sending LiDAR point clouds to deep networks for perception and planning, could the motion compensation consequently become a wide-open backdoor in those networks, due to both the adversarial vulnerability of deep learning and GPS-based vehicle trajectory estimation that is susceptible to wireless spoofing? We demonstrate such possibilities for the first time: instead of directly attacking point cloud coordinates which requires tampering with the raw LiDAR readings, only adversarial spoofing of a self-driving car's trajectory with small perturbations is enough to make safety-critical objects undetectable or detected with incorrect positions. Moreover, polynomial trajectory perturbation is developed to achieve a temporally-smooth and highly-imperceptible attack. Extensive experiments on 3D object detection have shown that such attacks not only lower the performance of the state-of-the-art detectors effectively, but also transfer to other detectors, raising a red flag for the community. The code is available on https://ai4ce.github.io/FLAT/.

1. Introduction

Autonomous driving systems are generally equipped with all kinds of sensors to perceive the complex environment [7]. Among the sensors, LiDAR has played a crucial role due to its plentiful geometric information sampled by incessantly spinning a set of laser emitters and receivers. However, LiDAR scans are easily distorted by vehicle's motion, *i.e.*, the points in a full sweep are sampled at different timestamps when vehicle is at different locations and orientations, as shown in Fig. 1. Imagine that a self-driving car is on a highway at a speed of 30m/s, its LiDAR with a 20Hz scanning frequency would move 1.5 meters during a full sweep, severely distorting the captured point cloud.

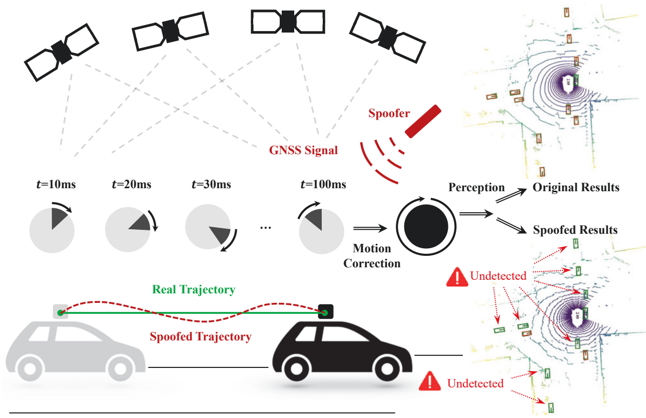


Figure 1: Illustration of our trajectory-based attack and the motion correction process. The top right and bottom right figures are respectively original and distorted LiDAR sweep as well as the detection results. **Green/red** boxes denote the ground truth/prediction. In this example, our method (named as FLAT) makes the detector miss eight out of eleven vehicles.

Such distortions are typically compensated by querying the vehicle/LiDAR's pose at any time from a continuous vehicle ego-motion tracking module that fuses pose estimation from Global Navigation Satellite System (GNSS, *e.g.*, GPS, GLONASS, and BeiDou), Inertial Navigation System (INS), and SLAM-based localization using LiDAR or cameras. Well-known LiDAR datasets like KITTI [7] and nuScenes [2] have already corrected such motion distortions prior to the dataset release. Researchers then made impressive progresses by processing those distortion-free point clouds using deep neural networks (DNNs) for many tasks, *e.g.*, 3D object detection, semantic/instance segmentation, motion prediction, multiple object tracking.

However, using DNNs on LiDAR point clouds creates a potentially dangerous and less cognizant vulnerability in self-driving systems. First, the above perception tasks are functions of LiDAR point clouds implemented via DNNs. Second, the motion compensation makes LiDAR point clouds a function of the vehicle trajectories. This functional composition leads to a simple but surprising fact that *those perception tasks are now also functions of the trajectories*. Thus, such a connection exposes the well-known adversarial robustness issue of DNNs to

* indicates equal contribution. The work is done during Congcong's visit at NYU. Chen Feng is the corresponding author.

hackers who could now fool a self-driving car's safety-critical LiDAR-depending perception modules by calculatedly spoofing the area's wireless GNSS signals, which is still a serious and unresolved security problem demonstrated on many practical systems [23, 30, 52]. Luckily, given that the aforementioned non-GNSS pose sources such as INS and visual SLAM are fused together for vehicle ego-motion estimation, large variations (meter-level) of GNSS trajectory spoofing could be detected and filtered, ensuring a safe localization and mapping for self-driving cars. However, *what if a spoofed trajectory only has dozens of centimeters offset? Would current point cloud DNNs be robust enough under such small variations?*

In this paper, we initiate the first work to reveal and investigate such a vulnerability. Different from existing works directly attacking point cloud coordinates with 3D point perturbations or adversarial object generation [9, 36, 41, 44], we propose to fool LiDAR-based perception modules by attacking the vehicle trajectory, which could be detrimental and easily deployable in the physical world. Our investigation includes how to obtain LiDAR sweeps with simulated motion distortion from real-world datasets, convert LiDAR point clouds as a differentiable function of the vehicle trajectories, and eventually calculate the adversarial trajectory perturbation and make them less perceptible. Our principal contributions are as follows:

- We propose an effective approach for simulating motion distortion using a sequence of real-world LiDAR sweeps from existing dataset.
- We propose a novel view of LiDAR point clouds as a differentiable function of the vehicle trajectories, based on the real-world motion compensation process.
- We propose to **F**ool **L**iDAR perception with **A**dversarial **T**rajectory (**FLAT**), which has better feasibility and transferability.
- We conduct extensive experiments on 3D object detection as a downstream task example, and show that the advanced detectors can be effectively blinded.

2. Related Work

GNSS/INS and LiDAR Motion Compensation. Motion distortion is also known as motion blur or rolling shutter effect of LiDAR on ego-motion vehicles [20]. To compensate for such distortions, GNSS/INS is often used to provide the pose of the LiDAR at any moment when a point is scanned. This opens backdoors for a self-driving system. First, space weather is a substantial error source for GNSS and also significantly affects systems such as differential GPS. The influence of ionosphere disturbances on GPS kinematic precise point positioning (PPP) can be larger than 2 to 10 meters at different latitudes [49], while solar radio burst could cause GPS positioning errors as large as 300 meters vertically and 50 meters horizontally [22]. Besides those naturally occurring events, malicious attacks such as GPS jamming or spoofing could also be used to arbitrarily modify the GPS trajectory [23, 30, 52]. Moreover, when GNSS and INS are fused, such attacks can affect not only positions but also the rotational component (gyroscope bias compensation) of a vehicle's trajectory [46]. Of course, nowadays for motion compensation, LiDAR poses are usually fused between GNSS/INS and LiDAR-/camera-based localization, so large pose variations from extreme space weather or "urban canyon" effect could be filtered. *But as long as GNSS is a part of the equation, the backdoors could remain open, especially when the spoofed trajectories only have small variations from the ground truth*, as we show later.

Image-based Adversarial Attack. Despite the great success achieved by deep learning in both academia and industry, researchers have found that deep networks are susceptible to carefully-designed adversarial perturbation which is hard to distinguish. Since such vulnerability is firstly pointed out in image classification task [35], broad attentions have been paid to adversarial robustness in various downstream tasks, *e.g.*, semantic segmentation [45], object detection [24], visual tracking [8], *etc*. Adversarial attack is divided into white box [21] and black box [1] based on whether the model parameters are known. Besides, attacks can be categorized as targeted [47] and untargeted [42] according to whether the adversary has a particular goal. Meanwhile, many attempts have been made in defense mechanisms, such as adversarial training [19], certified defense [28, 51], adversarial example detector [51], and ensemble diversity [25]. Extensive studies in image-based adversarial attack and defense have largely promoted the development of trustworthy machine learning in 2D computer vision and inspired similar investigations in 3D vision.

Point Cloud Attack and Defense. Recently, researchers have explored the vulnerability of DNNs taking point clouds of 3D objects as input. For the object-level point cloud attack, Xiang et al. [44] proposed point perturbation as well as cluster generation to attack the widely-used PointNet [26]. Besides, critical points removal [41, 48], adversarial deformation [54], geometric-level attack [15] are proposed for fooling the point cloud-based deep model. However, none of them directly target LiDAR point clouds of self-driving scenes which have domain gaps than object-level point clouds. Moreover, to implement the above point cloud attacks towards a self-driving car requires tampering with its software for altering point cloud coordinates. Differently, our paper reveals *a simple yet dangerous possibility* of spoofing the trajectory to attack deep modules through the LiDAR motion correction process yet *without any need of software hacking*. As for the works on scene-level point cloud attack, Tu et al. [36] proposed to generate 3D adversarial shapes placed on the rooftop of a target vehicle, making the target invisible for the detectors. Some works [3, 4, 32, 34] created spoofing obstacles/faked points in front of the car to influence the vehicle's decision, but they can only modify the limited area of the scene. Differently, our method does not need to physically alter any shapes in the scene, and can *easily scale up the attack to the whole scene*. In a word, existing works directly manipulate on the point coordinates either physically or virtually, while our attack is realized by spoofing the vehicle trajectories.

Affected Downstream Tasks. Theoretically, every downstream task that require LiDAR point cloud as input could be affected by the attacks proposed in this paper. This could include geometric vision tasks such as registration, pose estimation, and mapping, as well as pattern recognition tasks such as 3D object detection [14, 16, 31, 38, 50, 55], semantic segmentation [12, 13], motion prediction [17, 43, 53], and multiple object tracking [39, 40]. While the first group of tasks could be less severely affected by GNSS spoofing via data fusion as mentioned above, the second group has higher vulnerability, because small but calculated perturbations in the point coordinates could affect deep networks as demonstrated in the above related works. In this paper, without loss of generality, we choose to focus on the 3D object detection task to illustrate the severity of this backdoor, because miss detection of safety-critical objects surrounding a self-driving car could be a matter of life or death.

3. Motion Distortion in LiDAR

LiDAR measurements are obtained along with the rotation of its beams, so the measurements in a full sweep are captured at different timestamps, introducing motion distortion which jeopardizes the vehicle perception. Autonomous systems generally utilize LiDAR's location and orientation obtained from the localization system to correct distortion [6, 10, 29]. Most LiDAR-based datasets [2, 7] have finished synchronization before release. Hence, the performance of current 3D perception algorithms in the distorted point cloud remains unexplored. We briefly introduce the nomenclature in this work before detailed illustrations.

World Frame. We use a coordinate frame W fixed in the world with the orthonormal basis $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$ to describe the global displacement of the self-driving car.

Object Frame. The car can be associated with a right-handed, orthonormal coordinate frame which can describe its rigid body motion. Such a frame attached to the car is called object frame [18]. In the following, we use *frame* to denote the object frame of the car at different timestamps.

Sweep&Packet. LiDAR points in a complete 360° is called a *sweep*, and the points are emitted as a stream of *packets*, each covering a sector of the 360° coverage [10].

3.1. Linear Pose Interpolation

In this section, we recover the raw point cloud before synchronization with a sequence of real-world LiDAR point

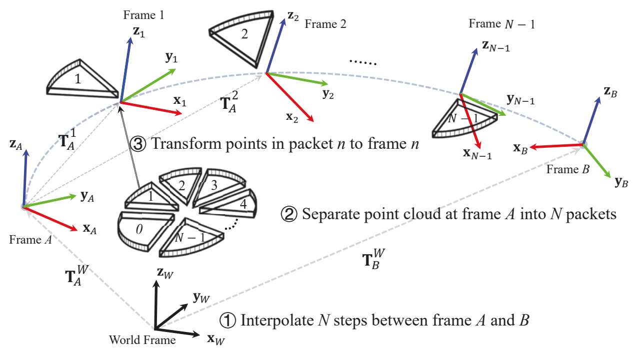


Figure 2: Diagram of motion distortion simulation. Firstly, we interpolate N steps in 6DoF pose between two adjacent frames A and B. Secondly, we divide a sweep at frame A into N packets, and the n-th sector corresponds to the n-th interpolated frame. Thirdly, we transform the point cloud in the n-th packet at frame A into frame n with homogeneous transformation $\mathbf{T}_A^n$. Finally, motion-distorted point cloud can be generated by aggregating the point cloud from frame 0 to frame $N-1$.

clouds from nuScenes [2]. Each sample of nuScenes provides a sweep and its corresponding ego pose. We assume the vehicle is moving smoothly and carry out linear pose interpolation between two adjacent frames A and B, as shown in Fig. 2. $\{\mathbf{x}_A, \mathbf{y}_A, \mathbf{z}_A\}$ and $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ are their orthonormal basis. Using $\mathbf{t}_A \in \mathbb{R}^3$ and $\mathbf{t}_B \in \mathbb{R}^3$ to represent global translation of frame A and B, the global translation of the n-th ($n = 0, 1, 2, ..., N-1$) interpolated frame is:

$$\mathbf{t}(n) = \mathbf{t}_A + \frac{\mathbf{t}_B - \mathbf{t}_A}{N} \times n \,, \tag{1}$$

where N is the total interpolation steps. For the orientation, we implement spherical linear interpolation (slerp) [33]. Using $\mathbf{q}_A = \left[q_w^A, q_x^A, q_y^A, q_z^A\right]^\top$ and $\mathbf{q}_B = \left[q_w^B, q_x^B, q_y^B, q_z^B\right]^\top$ to represent quaternion at frame A and B, then the quaternion at the n-th interpolated frame is:

$$\mathbf{q}(n) = \frac{\sin((1-n)\theta)}{\sin\theta}\mathbf{q}_A + \frac{\sin(n\theta)}{\sin\theta}\mathbf{q}_B \,, \tag{2}$$

where $\theta = \cos^{-1}(\mathbf{q}_A \cdot \mathbf{q}_B)$ is the rotation angle between A and B. Convert the quaternion $\mathbf{q}(n) = \left[q_w^n, q_x^n, q_y^n, q_z^n\right]^\top$ to the rotation matrix $\mathbf{R}(n) \in SO(3)$, then the homogeneous transformation matrix from the world frame W to the n-th interpolated frame denoted as $\mathbf{T}_n^W \in SE(3)$ is:

$$\mathbf{T}_n^W = \begin{bmatrix} \mathbf{R}(n) & \mathbf{t}(n) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} . \tag{3}$$

3.2. Motion Distortion Simulation

After linear pose interpolation, we can get N more frames. To simulate motion distortion, we assume that the sweep at frame A is aggregated from N frames, and in each frame, the beam scans a degree of $\frac{360}{N}$. Therefore, we firstly divide the sweep at frame A into N packets, as shown in Fig. 2. Then we transform the LiDAR packet n to the coordinate of frame n at which the point cloud is assumed to be captured. The points in packet n at frame A (denoted as ${}^{A}\mathbf{P}_n \in \mathbb{R}^{4\times m_n}$) is represented as a set of 3D points $\{P_m | m = 1, 2, ..., m_n\}$, where each point P_m is a vector of its homogeneous coordinate $(x, y, z, 1)$, and m_n denotes the number of points in packet n. We transform ${}^{A}\mathbf{P}_n$ to its corresponding capture frame n:

$$ {}^{n}\mathbf{P} = \mathbf{T}_A^n \, {}^{A}\mathbf{P}_n \,, \tag{4} $$

where ${}^{n}\mathbf{P} \in \mathbb{R}^{4\times m_n}$ is the point cloud of packet n in the coordinate of frame n (can be treated as the point cloud captured at frame n) and $\mathbf{T}_A^n$ is the homogeneous transformation from frame n to frame A and is as follows:

$$ \mathbf{T}_A^n = \mathbf{T}_W^n \mathbf{T}_A^W = (\mathbf{T}_n^W)^{-1} \mathbf{T}_A^W \,, \tag{5} $$

where $\mathbf{T}_n^W$ is from Eq. (3) and $\mathbf{T}_A^W$ can also be calculated given $\mathbf{t}_A$ and $\mathbf{q}_A$. Finally, a set of motion-distorted point clouds ${}^{A}\widetilde{\mathbf{P}} \in \mathbb{R}^{4\times m}$ is generated by aggregating multiple LiDAR packets captured at different frames:

$$ {}^{A}\widetilde{\mathbf{P}} = [{}^{0}\mathbf{P}; {}^{1}\mathbf{P}; ...; {}^{N-1}\mathbf{P}] \,, \tag{6} $$

where $[\cdot;\cdot;...;\cdot]$ is the concatenation operation along the row. It is noted that $m = \sum_{n=0}^{N-1} m_n$.

3.3. Motion Compensation with Ego-Pose

Up to now, we have generated motion-distorted point cloud as shown in Eq. 6, and then we can represent the motion compensation using the mathematical formula, *i.e.*, a LiDAR sweep can be written as a differentiable function of the vehicle trajectory*. The undistorted point cloud ${}^{A}\mathbf{P} \in \mathbb{R}^{4\times m}$ is obtained by transforming point clouds from frame $0 \sim N-1$ back to the coordinate of frame A:

$$ {}^{A}\mathbf{P} = [\mathbf{T}_0^A \, {}^{0}\mathbf{P}; \mathbf{T}_1^A \, {}^{1}\mathbf{P}; ...; \mathbf{T}_{N-1}^A \, {}^{N-1}\mathbf{P}] \,, \tag{7} $$

where $\mathbf{T}_n^A$ is the transformation from frame A to frame n, which is the inverse matrix of $\mathbf{T}_A^n$ in Eq. (5).

4. Adversarial Trajectory Perturbation

4.1. Point Cloud Representation w.r.t. Trajectory

The studies of object-level point cloud generally treat point cloud as a set of 3D points sampled from mesh models instantaneously [9, 44]. In autonomous driving, however, the 3D points are captured in a dynamic setting through raycasting, thus LiDAR not only records points' (x, y, z) coordinates, but also the timestamps at which the points are captured. To this end, we propose a novel representation of point cloud as a function of vehicle trajectory through Eq. (7) which can be written as a general form:

$$ \mathbf{P} = f(\mathbf{T}, L({}^{n}\mathbf{P})), \;\; n = 0, 1, ..., N-1 \,, \tag{8} $$

*In this work, the translation and the orientation are collectively referred to as "trajectory".

where $\mathbf{P} \in \mathbb{R}^{4\times m}$ is a full sweep with m points, and is represented as a differentiable function f in terms of vehicle trajectory $\mathbf{T} \in \mathbb{R}^{N\times 4\times 4}$ and the list of LiDAR packets $L({}^{n}\mathbf{P}) = [{}^{0}\mathbf{P}, {}^{1}\mathbf{P}, ..., {}^{N-1}\mathbf{P}]$ (${}^{n}\mathbf{P} \in \mathbb{R}^{4\times m_n}$). Noted that the trajectory between two adjacent frames is represented as a set of homogeneous transformation matrices. Different from previous works tampering with point coordinates [44]†, we propose to represent point cloud as a function of trajectory, and attack the trajectory instead of 3D points. Our method has the following advantages.

Physical Feasibility. Since the motion compensation is naturally occurring in self-driving, it is physically-realizable and straightforward to attack the trajectory, *e.g.*, by wireless GNSS spoofing. In contrast, coordinate attack requires software hacking which is infeasible in practice.

Better Transferability. Our learned trajectory perturbation with the same size ($N \times 4 \times 4$) can be easily transferred to different sweeps. Yet, coordinate attacks cannot be transferred across sweeps because different sweeps could have different numbers of points, and the perturbation in the point space could have different dimensions.

Novel Parameterization. We attack the 6-DoF pose of each packet, but coordinate attack modifies a single point's xyz position without orientations. Hence, our method has better performance due to new attacked parameters.

4.2. Objective Function

Since the point cloud is represented as a differentiable function of the trajectory, the gradient can be back-propagated to the trajectory smoothly for adversarial learning. The adversarial objective is to minimize the negative loss function of the deep model with parameter θ:

$$ \min \widetilde{\mathcal{L}}(\theta, f(\mathbf{T}, L({}^{n}\mathbf{P})), \mathbf{y}) \,, \tag{9} $$

where $\mathbf{y}$ is the desired output of the network, $f(\mathbf{T}, L({}^{n}\mathbf{P}))$ is the input point cloud as a function of the trajectory. Assume that we are assigned to attack a downstream task with loss function $\mathcal{L}$, *e.g.*, cross-entropy loss in classification, then $\widetilde{\mathcal{L}} = -\mathcal{L}$. Afterwards, the adversarial perturbation $\boldsymbol{\delta} \in \mathbb{R}^{N\times 4\times 4}$ can be obtained via projected gradient descent (PGD) with multiple iterations [19]:

$$ \boldsymbol{\delta}^{t+1} = \mathcal{P}\left(\boldsymbol{\delta}^{t} - \alpha \,\mathrm{sgn}\left(\nabla_{\boldsymbol{\delta}} \widetilde{\mathcal{L}}(\theta, f(\mathbf{T}+\boldsymbol{\delta}, L({}^{n}\mathbf{P})), \mathbf{y})\right)\right) \,, \tag{10} $$

where t denotes iteration number, α denotes learning rate, $\mathcal{P}$ indicates the projection onto the convex set of interest, and this work uses clipping which is the case of ℓ_∞ norm to ensure an acceptable perturbation magnitude. The n-th element in $\boldsymbol{\delta}$ in the matrix form is:

$$ \boldsymbol{\delta}(n) = \begin{bmatrix} \widetilde{\mathbf{R}}(n) & \widetilde{\mathbf{t}}(n) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} . \tag{11} $$

†In the following, we will use *coordinate attack* to denote the modification of point coordinate.

Polynomial Trajectory Perturbation. To make the attack highly imperceptible, the polynomial trajectory perturbation is proposed: we define the perturbation in translations as a third-order polynomial of time as follows:

$$\widetilde{\mathbf{t}}(n) = \boldsymbol{\beta}^T \mathbf{n} \,, \tag{12}$$

where $\mathbf{n} = [1, n, n^2, n^3]^\top$ and $\boldsymbol{\beta} = [\boldsymbol{\beta_x}, \boldsymbol{\beta_y}, \boldsymbol{\beta_z}]$ is the polynomial coefficients. Now $\boldsymbol{\delta}$ is a differentiable function of $\boldsymbol{\beta}$, and the gradient will be back-propagated to $\boldsymbol{\beta}$, so the adversarial coefficients are calculated by:

$$\boldsymbol{\beta}^{t+1} = \mathcal{P}\left(\boldsymbol{\beta}^t - \alpha\,\mathrm{sgn}\left(\nabla_{\boldsymbol{\beta}}\widetilde{\mathcal{L}}(\theta, f(\mathbf{T}+\boldsymbol{\delta}, L({}^n\mathbf{P})), \mathbf{y})\right)\right) \,, \tag{13}$$

hence, we only need to *manipulate several key points to bend a polynomial-parameterized trajectory* which can be easily achieved in reality, achieving a real-time attack.

4.3. Attack with Regularization

Regularization on Trajectory. To realize an imperceptible attack, we propose a trajectory smoothness regularizer to repress the total variation in vehicle poses. Given a trajectory perturbation $\boldsymbol{\delta} \in \mathbb{R}^{N\times 4\times 4}$, we calculate the difference in translation and rotation separately. The smoothness $\mathcal{S}(\boldsymbol{\delta})$ equals to the following formula:

$$\lambda_{\mathbf{t}}(\sum_{n=1}^{N-1}(\widetilde{\mathbf{t}}(n)-\widetilde{\mathbf{t}}(n-1))^p)^{\frac{1}{p}} + \lambda_{\mathbf{R}}(\sum_{n=1}^{N-1}(\widetilde{\mathbf{R}}(n)-\widetilde{\mathbf{R}}(n-1))^p)^{\frac{1}{p}} \,, \tag{14}$$

where $\lambda_{\mathbf{t}}$ and $\lambda_{\mathbf{R}}$ are used to balance the influence of translations and rotations, p denotes the norm ($p = 2$ in this work). With trajectory smoothness regularization, the objective of adversarial attacks is as follows:

$$\min \widetilde{\mathcal{L}}(\theta, f(\mathbf{T}+\boldsymbol{\delta}, L({}^n\mathbf{P})), \mathbf{y}) + \lambda_s \mathcal{S}(\boldsymbol{\delta}) \,, \tag{15}$$

where λ_s is for adjusting the smoothness degree. By optimizing over Eq. (15), we aim to find an imperceptible adversarial perturbation $\boldsymbol{\delta}$ with desirable smoothness.

Regularization on Point Cloud. We also propose a regularizer on point cloud to repress its change. We use two metrics to measure the variation of point cloud before and after distortion, *i.e.*, ℓ_p norm and Chamfer distance [5]. Using $\mathcal{D}_L(\boldsymbol{\delta})$ to denote ℓ_p distance, $\mathcal{D}_C(\boldsymbol{\delta})$ to denote Chamfer distance before and after perturbation $\boldsymbol{\delta}$. The adversarial objective with regularization is defined as:

$$\min \widetilde{\mathcal{L}}(\theta, f(\mathbf{T}+\boldsymbol{\delta}, L({}^n\mathbf{P})), \mathbf{y}) + \lambda_d \mathcal{D}(\boldsymbol{\delta}) \,, \tag{16}$$

where $\mathcal{D}(\boldsymbol{\delta})$ can be either $\mathcal{D}_L(\boldsymbol{\delta})$ or $\mathcal{D}_C(\boldsymbol{\delta})$. λ_d is to control the degree of distortion. By optimizing over Eq. (16), we try to search for a powerful adversarial perturbation $\boldsymbol{\delta}$ leading to subtle distortion in the point cloud.

5. Experiments

5.1. Target 3D Deep Model

In this work, we select widely-studied LiDAR-based 3D detection, which aims to estimate 3D bounding boxes of the objects in point cloud, as a downstream task example to verify our attack pipeline. Currently, LiDAR-based detection has two mainstreams: 1) point-based method directly consuming raw point cloud data, 2) voxel-based method which requires non-differentiable voxelization in the preprocessing stage. For the white box attack, we use point-based PointRCNN [31]. For the black box transferability test, we adopt voxel-based PointPillar++ [11].

PointRCNN. Our white box model, PointRCNN, uses PointNet++ [27] as its backbone and includes two stages: stage-1 for proposal generation based on each foreground point, and stage-2 for proposal refinement in the canonical coordinate. Since PointRCNN uses raw point cloud as the input, the gradient can smoothly reach the point cloud, then arrive at vehicle trajectory. In this work, we individually attack the classification as well as regression branches in stage-1 and stage-2, with four attack targets in total.

PointPillar++. PointPillar [14] proposes a fast point cloud encoder using pseudo-image representation. It divides point cloud into bins and uses PointNet [26] to extract the feature for each pillar. Due to the non-differentiable preprocesssing stage, the gradient cannot reach the point cloud. Peiyun *et al.* [11] proposed to augment PointPillar with the visibility map, achieving better precision. In this work, we use PointPillar++ to denote PointPillar with visibility map in [11]. We use perturbation learned from the white box PointRCNN to attack black box PointPillar++, in order to examine the transferability of our attack pipeline.

5.2. Dataset and Evaluation Metrics

Dataset. nuScenes [2] is a large-scale multimodel autonomous driving dataset captured by a real SDV with a full 360° field of view in various challenging urban driving scenarios. Including 1000 scenes collected in Boston and Singapore in different weather, nuScenes has much more annotations (7 times) and images (100 times) than the pioneer KITTI [7]. Besides, nuScenes provides a temporal sequence of samples in each scene, facilitating linear pose interpolation for motion distortion simulation, yet the 3D detection dataset in KITTI solely offers independent frames without temporal connection. Considering the above factors, nuScenes is employed in this work. We use PointRCNN model released in [37], and the open-source PointPillar++ model in [11]. Both models are trained on nuScenes training set. We report the results of white box on 1,000 samples from the validation set, and the results of black box on the whole validation set.

Metrics. For the white box PointRCNN, we report the

Table 1: **Quantitative results of white box attack:** AP (IoU=0.7) of 3D bounding boxes on nuScenes [2]. We report results of the car category under different levels of difficulty and ranges of depth following [37]. In four attack settings of our method (**FLAT**), the best and second best attack qualities among four attacking targets are respectively highlighted using **red** and **blue** color. In attacking translations/rotations, the step size for each iteration is 0.1 and 0.01 respectively, and the number of attack iteration is 20 for both two settings.

Attack Approach \ Case			Easy	Moderate	Hard	0-30m	30-50m	50-70m
PointRCNN [31]			47.44	21.56	20.91	47.44	2.16	0.17
Baseline	Coordinate Attack [44]		16.42	6.58	5.90	15.20	0.48	0.03
	Random Attack (Point Cloud)		30.09	12.39	10.84	25.65	0.79	0.06
	Random Attack (Translation)		17.00	8.58	8.90	20.43	1.09	0.09
	Random Attack (Rotation)		12.30	4.87	5.13	13.31	0.01	0.00
	Random Attack (Full Trajectory)		5.66	2.43	2.78	7.67	0.02	0.00
FLAT (Translation)	Classification	Stage-1	**12.94**	**6.58**	**7.22**	**16.82**	**0.86**	0.06
		Stage-2	**11.72**	**5.87**	**6.06**	**13.91**	**0.87**	**0.04**
	Regression	Stage-1	17.46	8.24	8.57	19.36	1.09	**0.03**
		Stage-2	26.09	12.78	12.53	27.15	2.17	0.17
FLAT (Polynomial)	Classification	Stage-1	**17.94**	**9.36**	**9.56**	**20.73**	1.90	0.19
		Stage-2	**12.51**	**6.37**	**6.54**	**14.51**	**1.38**	**0.16**
	Regression	Stage-1	22.60	11.01	10.96	24.36	**1.67**	**0.15**
		Stage-2	26.04	12.76	12.51	27.19	2.17	0.17
FLAT (Rotation)	Classification	Stage-1	6.32	2.43	2.51	7.32	**0.02**	**0.00**
		Stage-2	**2.35**	**0.80**	**0.61**	**2.03**	**0.01**	**0.00**
	Regression	Stage-1	**5.50**	**1.87**	**1.76**	**5.45**	**0.02**	**0.00**
		Stage-2	26.30	12.89	12.59	27.35	2.17	**0.17**
FLAT (Full Trajectory)	Classification	Stage-1	1.52	0.45	0.51	1.71	**0.01**	**0.00**
		Stage-2	**0.19**	**0.01**	**0.02**	**0.26**	**0.00**	**0.00**
	Regression	Stage-1	**1.01**	**0.35**	**0.32**	**1.27**	**0.01**	**0.00**
		Stage-2	26.03	12.70	12.48	27.13	2.16	**0.17**

3D bounding boxes average precision (AP) with the IoU thresholds at 0.7 on car category. Following [37], we evaluate the detector in three scenarios (Easy, Moderate, and Hard) based on the difficulty level of the surrounding cars. In addition, the performance within three depth ranges, *i.e.*, $0 \sim 30$, $30 \sim 50$, and $50 \sim 70$ meters, are also assessed. For the black box PointPillar++, we follow the original paper [11] to employ official 3D detection evaluation protocol in nuScenes [2], *i.e.*, average mAP over ten categories at four distance thresholds. For evaluating attack quality, we utilize the performance drop after attack.

5.3. Experimental Setup

Implementation details. nuScenes samples keyframes at 2Hz from the original 20Hz data, so we assume that a sweep consumes 0.5 second[‡] and implement linear pose interpolation between two adjacent keyframes. We set total interpolation step N as 100. For PGD, we restrict the perturbation to 10cm in translation, and to 0.01 in rotation. The step size for each attack iteration is 0.1/0.01 in translation/rotation, and the number of iterations is 20. More experimental settings like the step size and the number of iterations are reported in the supplementary.

Baselines. To demonstrate the superiority of our attack pipeline, we employ two baseline methods for comparison.

- **Random Attack.** We add Gaussian noise with a 10cm standard deviation to the original point cloud, Gaussian noise with a 10cm standard deviation to the translations and Gaussian noise with 0.01 standard deviation to rotations.
- **Coordinate Attack.** Coordinate attack [44] is directly manipulating the point set to deceive the detector. We attack classification of stage-2 and restrict the change in point coordinate to 10cm. The binary search step number is 10 and the number of iteration for each binary search step is 100.

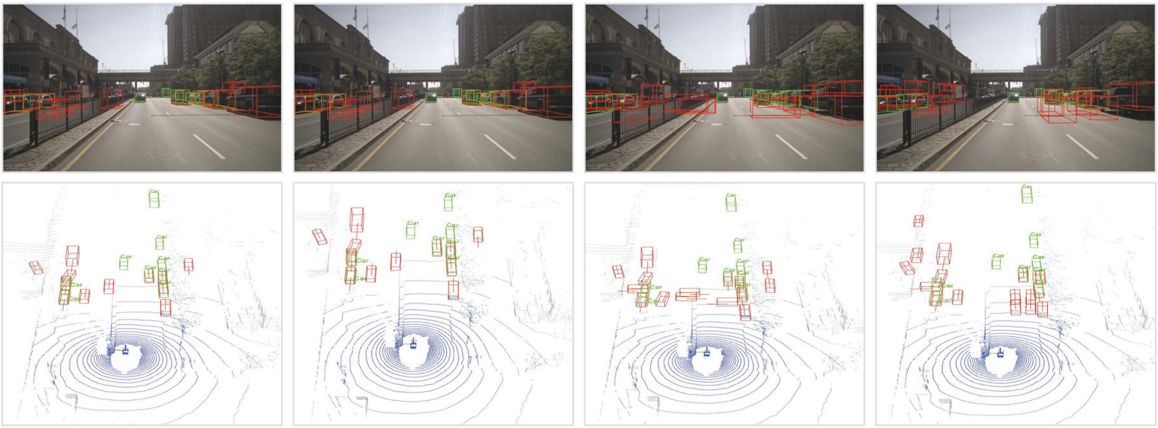
Attack Settings. The polynomial trajectory attack is temporally-smooth and the others are temporally-discrete:

- **Attack Translation Only.** We merely modify the translation vector. The perturbation is a 100×3 matrix.
- **Polynomial Perturbation.** We add a polynomial perturbation into the vehicle trajectory (translation part).
- **Attack Rotation Only.** We only interfere with the rotation matrix. The perturbation is a $100 \times 3 \times 3$ tensor.
- **Attack Full Trajectory.** We tamper with the transformation matrix. The perturbation is a $100 \times 4 \times 4$ tensor.

5.4. White Box Attack

To explore the vulnerability of different stages and branches in PointRCNN, we separately attack its four modules, *i.e.*, classification/regression branch of stage-1/2. Qualitative results are displayed in Fig. 3 and more examples can be found in the supplementary.

[‡]In fact, the LiDAR rotation period is 0.05 second, however, we only have access to annotations on keyframes at 2Hz, so we have to assume the capture frequency of LiDAR is also 2Hz.



a) PointRCNN b) FLAT (Translation) c) FLAT (Rotation) d) FLAT (Full Trajectory)

Figure 3: Qualitative evaluations of white box attack. False positives are increased and output are drifted by carefully crafting the trajectory.

Table 2: Quantitative results of black box attack on nuScenes [2]: the worst and second worst performances are highlighted by **red** and **blue** color.

Category	Car	Pedes.	Barri.	Traff.	Truck	Bus	Trail.	Const.	Motor.	Bicyc.	mAP
PointPillar++ [11]	80.0	66.9	34.5	27.9	35.8	54.1	28.5	7.5	18.5	**0.0**	35.4
FLAT (Translation)	57.7	26.9	21.3	12.6	25.3	30.0	25.0	3.3	3.5	**0.0**	20.6
FLAT (Polynomial)	57.9	26.9	21.3	12.7	25.4	30.2	25.2	3.4	3.5	**0.0**	20.7
FLAT (Rotation)	**47.7**	**21.1**	**18.6**	**8.9**	**20.2**	**23.5**	**20.5**	**1.7**	**1.3**	**0.0**	**16.4**
FLAT (Full Trajectory)	**45.0**	**18.6**	**16.4**	**5.9**	**18.3**	**22.0**	**19.7**	**1.3**	**0.5**	**0.0**	**14.8**

Attack Translation Only. The quantitative results are exhibited in Table 1. Simply adding random noise with a small standard deviation can largely decrease the performance, *e.g.*, AP in the easy case is reduced by 30.44 (around 64%). This phenomenon should stand as a warning for the autonomous driving community. Also, attacking classification branch of stage-2 is the most effective way to fool the detector: AP is respectively lowered by 35.72 (75.3%), 15.69 (72.8%) and 14.85 (71.0%) in the scenarios of easy, moderate and hard. This is because the stage-2 outputs the final predictions and is extremely safety-relevant. Moreover, we can find that attacking the classification branch is more effective than attacking regression, which is reasonable because classifying the objects takes priority over estimating their sizes in the detection task. Compared to the random attack, our method is more detrimental thanks to our making use of the vulnerability pointed out by the gradient. In easy, moderate, and hard scenarios, adversarial perturbation generated by attacking stage-2's classification can yield additional drops of 5.28, 2.71, and 2.84 in AP, compared to the random perturbation. Besides, our attack is better than the coordinate attack due to the parameterization, validating the superiority of the trajectory attack.

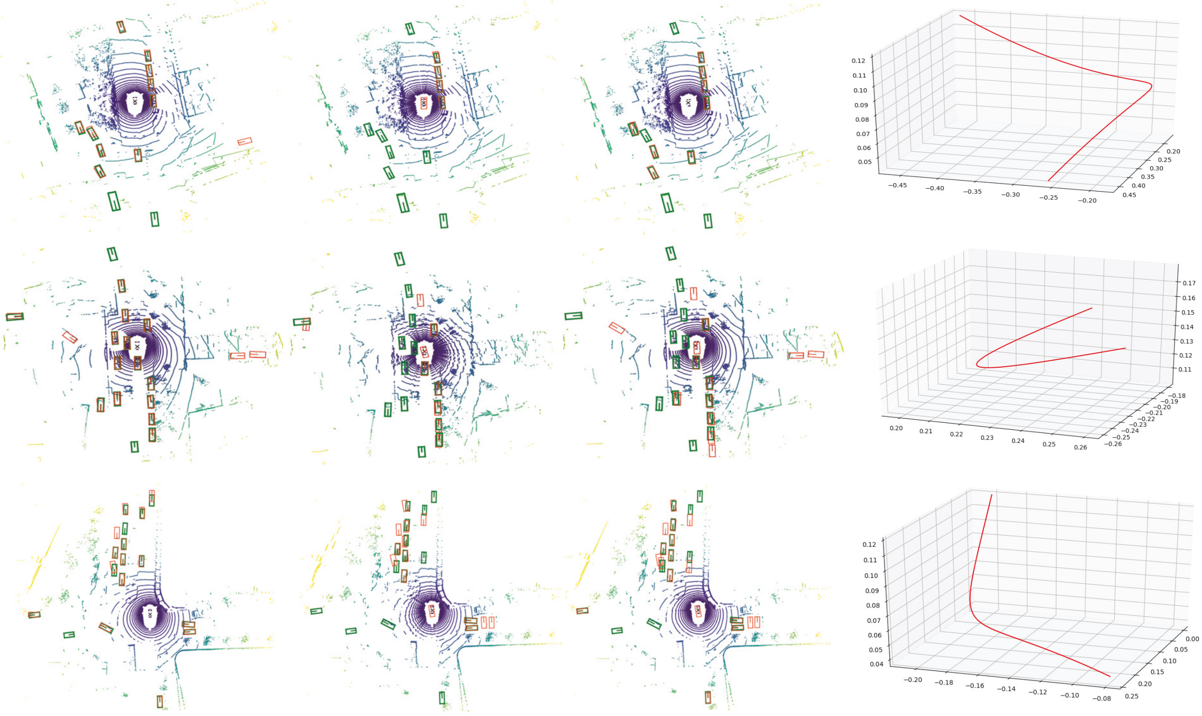
Polynomial Trajectory Perturbation. When attacking polynomial coefficients instead of the individual trajectory point, the performance is still on par with the discrete setting, as shown in Table 1, yet the attack is highly imperceptible especially in the point cloud space as shown in Fig. 4.

Table 3: AP of FLAT with and without regularization. λ indicates regularization strength $\mathcal{S}$ denotes average trajectory variation and $\mathcal{D}$ denotes average point cloud distance (ℓ_p norm and Chamfer distance).

Regularization	λ	$\mathcal{S}/\mathcal{D}$	AP Easy	AP Moderate	AP Hard
Trajectory	0	0.18	0.19	0.01	0.02
	0.01	**0.17**	**0.28**	**0.04**	**0.02**
ℓ_p norm/Chamfer	0	27.54/28.70	0.19/0.19	0.01/0.01	0.02/0.02
	0.01	**14.43/13.85**	**5.66/3.11**	**2.45/1.33**	**2.65/1.38**
	0.1	2.35/3.54	21.84/14.59	9.98/6.45	9.75/ 6.59
	1	0.95/2.04	23.65/19.19	11.22/9.21	11.33/9.40

Attack Rotation Only. Coordinate attack is manipulating each point's xyz position. In contrast, our method treat each LiDAR packet as a rigid body, therefore we can attack the rotation. From Table 1 we can find that attacking classification in stage-2 is still the most powerful way to fool PointRCNN. Besides, attacking rotation achieves more performance drop compared to attacking translation, *e.g.*, in easy, moderate, and hard situations, AP is respectively decreased by 45.09 (95.0%), 20.76 (96.3%) and 20.30 (97.1%) in comparison with original PointRCNN. Moreover, fooling regression in stage-1 achieves the second best attacking quality, proving the effectiveness of attacking the fundamental proposal generation. Besides, attacking regression of stage-2 has the worst attacking quality, proving that attacking the refinement of box size has no significant effect. Compared to the random attack, attacking stage-2's classification has realized additional AP drop (9.95, 4.07, 4.52), validating the merits of adversarial learning.

Attack Full Trajectory. As shown in Table 1, fooling the full trajectory has achieved the best attacking quality,



a) Original Predictions b) FLAT (Full Trajectory) c) FLAT (Polynomial) d) Polynomial Trajectory Perturbation

Figure 4: Point cloud visualization and qualitative results of black box attack. a) Raw detections of the original detector PointPillar++ [11]. b) The output of the detector after attacking the full trajectory. c) The output of the detector after polynomial trajectory perturbation in the euclidean space. d) The polynomial translation perturbation visualized in xyz space, the units of three axes are all meters. **Green**/**red** boxes denote the ground truth/prediction.

e.g., while attacking classification of stage-2, AP can be decreased to nearly zero. Compared to original PointRCNN, AP is respectively lowered by 47.25 (99.6%) in easy scenarios, indicating that the detector is completely blinded.

Attack with Regularization. As shown in Table 3, for regularizing trajectory smoothness, average trajectory variation is slightly lowered and the performance is still on par with attacking without regularization. For regularizing point cloud change, when $\lambda = 0.01$, point distance is decreased by 47.6% (ℓ_p norm) and 51.7% (Chamfer distance), while the APs in three scenarios are still very low, thus an advanced imperceptible attack can be realized with regularization. Qualitative examples are in the supplementary.

5.5. Black Box Attack

We choose voxel-based PointPillar++ [11] to test the transferability across different input representation. The quantitative results on ten categories are shown in Table 2: the performance is still largely dropped by our attack, *e.g.*, mAP on ten categories can be reduced by 20.6 (58.2%), and the AP on car is decreased by 35.0 (43.8%). Meanwhile, our method has demonstrated satisfactory transferability across categories. With only adversarial learning against the car detector, the detection of other categories are also deceived. For small object like pedestrian/motorcycle, AP can be dropped by 48.3 (72.2%)/18.0 (97.3%), which has demonstrated the superior attacking quality of our method against small object detector. This superiority is mainly because the small object with less points is more susceptible to the perturbation compared to the large object like bus (dropped by 32.1, 59.3%) or truck (dropped by 17.5, 48.9%). Several qualitative examples are displayed in Fig. 4 and more examples are in the supplementary.

6. Conclusion

We proposed a generic and feasible DNN attack pipeline based on the trajectory against LiDAR perception. We conduct experiments on the well-studied 3D object detection task. In white box attack, even only with a 10cm perturbation in translations, the precision can be dropped by around 70%. While attacking the full trajectory, the precision can be decreased to nearly zero, yet the attack is less perceptible (especially the point clouds). Our attack also shows good transferability across various input representations and target categories, raising a red flag for perception systems using LiDAR and DNN jointly.

Acknowledgment. The research is supported by NSF FW-HTF program under DUE-2026479. The authors gratefully acknowledge the useful comments and suggestions from Yong Xiao, Wenxiao Wang, Chenzhuang Du, Wang Zhao, Ziyuan Huang, Hang Zhao and Siheng Chen, and also thank Yan Wang, Shaoshuai Shi and Peiyun Wu for their helpful open-source code.

References

[1] W. Brendel, Jonas Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representation*, 2018. 2

[2] H. Caesar, Varun Bankiti, A. Lang, Sourabh Vora, Venice Erin Liong, Q. Xu, A. Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11618–11628, 2020. 1, 3, 5, 6, 7

[3] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Q. Chen, K. Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2019. 3

[4] Yulong Cao, Chaowei Xiao, Dawei Yang, J. Fang, Ruigang Yang, M. Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *ArXiv*, abs/1907.05418, 2019. 3

[5] Haoqiang Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2463–2471, 2017. 5

[6] Davi Frossard, Simon Suo, S. Casas, J. Tu, Rui Hu, and R. Urtasun. Strobe: Streaming object detection from lidar packets. In *4th Conference on Robot Learning*, 2020. 3

[7] Andreas Geiger, Philip Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 1, 3, 5

[8] Q. Guo, Xiaofei Xie, Felix Juefei-Xu, L. Ma, Zhongguo Li, Wanli Xue, W. Feng, and Y. Liu. Spark: Spatial-aware online incremental attack against visual tracking. In *European Conference on Computer Vision*, 2020. 2

[9] Abdullah Hamdi, S. Rojas, A. Thabet, and Bernard Ghanem. Advpc: Transferable adversarial perturbations on 3d point clouds. In *European Conference on Computer Vision*, 2020. 2, 4

[10] Wei Han, Zhengdong Zhang, Benjamin Caine, B. Yang, C. Sprunk, O. Alsharif, J. Ngiam, V. Vasudevan, Jonathon Shlens, and Z. Chen. Streaming object detection for 3-d point clouds. In *European Conference on Computer Vision*, 2020. 3

[11] Peiyun Hu, Jason Ziglar, David Held, and D. Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10998–11006, 2020. 5, 6, 7, 8

[12] Q. Hu, B. Yang, Linhai Xie, S. Rosa, Yulan Guo, Zhihua Wang, A. Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11105–11114, 2020. 3

[13] L. Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4866–4875, 2020. 3

[14] A. Lang, Sourabh Vora, H. Caesar, Lubing Zhou, J. Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12689–12697, 2019. 3, 5

[15] Itai Lang, Uriel Kotlicki, and S. Avidan. Geometric adversarial attacks and defenses on 3d point clouds. *ArXiv*, abs/2012.05657, 2020. 2

[16] M. Liang, B. Yang, Yun Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7337–7345, 2019. 3

[17] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 3

[18] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012. 3

[19] A. Madry, Aleksandar Makelov, L. Schmidt, D. Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representation*, 2018. 2, 4

[20] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020. 2

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016. 2

[22] Bilal Muhammad, Valentina Alberti, Alessandro Marassi, Ernestina Cianca, and Mauro Messerotti. Performance assessment of gps receivers during the september 24, 2011 solar radio burst event. *Journal of Space Weather and Space Climate*, 5:A32, 2015. 2

[23] Sashank Narain, Aanjhan Ranganathan, and G. Noubir. Security of gps/ins based on-road location tracking systems. In *IEEE Symposium on Security and Privacy (SP)*, pages 587–601, 2019. 2

[24] Tianyu Pang, Chao Du, Y. Dong, and J. Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems*, 2018. 2

[25] Tianyu Pang, Kun Xu, Chao Du, N. Chen, and J. Zhu. Improving adversarial robustness via promoting ensemble diversity. In *ICML*, 2019. 2

[26] C. R. Qi, H. Su, Kaichun Mo, and L. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, 2017. 2, 5

[27] C. R. Qi, L. Yi, H. Su, and L. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 2017. 5

[28] Aditi Raghunathan, J. Steinhardt, and Percy Liang. Certi-

fied defenses against adversarial examples. In *International Conference on Learning Representation*, 2018. 2

[29] Mao Shan, J. Berrio, S. Worrall, and E. Nebot. Probabilistic egocentric motion correction of lidar point cloud and projection to camera images for moving platforms. In *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020. 3

[30] Junjie Shen, J. Won, Zeyuan Chen, and Q. Chen. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing (extended version). In *USENIX Security Symposium*, 2020. 2

[31] Shaoshuai Shi, X. Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 3, 5, 6

[32] H. Shin, Dohyun Kim, Yujin 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*, volume 2017, page 613, 2017. 3

[33] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH*, 1985. 3

[34] Jiachen Sun, Yulong Cao, Q. Chen, and Z. Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020. 3

[35] Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representation*, 2014. 2

[36] J. Tu, Mengye Ren, Siva Manivasagam, Ming Liang, Bin Yang, R. Du, Frank C Z Cheng, and R. Urtasun. Physically realizable adversarial examples for lidar object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13713–13722, 2020. 2

[37] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020. 5, 6

[38] Yue Wang, Alireza Fathi, Abhijit Kundu, D. Ross, C. Pantofaru, Tom Funkhouser, and J. Solomon. Pillar-based object detection for autonomous driving. In *European Conference on Computer Vision*, 2020. 3

[39] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE International Conference on Intelligent Robots and Systems*, 2020. 3

[40] Xinshuo Weng, Yongxin Wang, Y. Man, and Kris Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6498–6507, 2020. 3

[41] Matthew Wicker and M. Kwiatkowska. Robustness of 3d deep learning in an adversarial setting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11759–11767, 2019. 2

[42] Aming Wu, Yahong Han, Q. Zhang, and Xiaohui Kuang. Untargeted adversarial attack via expanding the semantic gap. In *Int. Conf. Multimedia and Expo*, pages 514–519, 2019. 2

[43] Pengxiang Wu, Siheng Chen, and D. Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11382–11392, 2020. 3

[44] Chong Xiang, Charles R. Qi, and B. Li. Generating 3d adversarial point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9128–9136, 2019. 2, 4, 6

[45] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision*, pages 1378–1387, 2017. 2

[46] Rui Xu, Mengyu Ding, Ya Qi, Shuai Yue, and Jianye Liu. Performance analysis of gnss/ins loosely coupled integration systems under spoofing attacks. *Sensors*, 18(12):4108, 2018. 2

[47] B. Yan, D. Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 987–996, 2020. 2

[48] Jiancheng Yang, Qiang Zhang, Rongyao Fang, B. Ni, J. Liu, and Q. Tian. Adversarial attack and defense on point sets. *ArXiv*, abs/1902.10899, 2019. 2

[49] Zhe Yang, YT Jade Morton, Irina Zakharenkova, Iurii Cherniak, Shuli Song, and Wei Li. Global view of ionospheric disturbance impacts on kinematic gps positioning solutions during the 2015 st. patrick's day storm. *Journal of Geophysical Research: Space Physics*, 125(7):e2019JA027681, 2020. 2

[50] Z. Yang, Y. Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11037–11045, 2020. 3

[51] Tao Yu, Shengyuan Hu, C. Guo, Wei-Lun Chao, and Kilian Q. Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In *Advances in Neural Information Processing Systems*, 2019. 2

[52] Kexiong Zeng, Shinan Liu, Y. Shu, Dong Wang, Haoyu Li, Yanzhi Dou, G. Wang, and Y. Yang. All your gps are belong to us: Towards stealthy manipulation of road navigation systems. In *USENIX Security Symposium*, 2018. 2

[53] Wenyuan Zeng, W. Luo, Simon Suo, A. Sadat, Bin Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8652–8661, 2019. 3

[54] Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and N. Yu. Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10353–10362, 2020. 2

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