Physically Realizable Adversarial Examples for LiDAR Object Detection

James Tu[1] Mengye Ren[1,2] Siva Manivasagam[1,2] Ming Liang[1]
Bin Yang[1,2] Richard Du[3] Frank Cheng[1,2] Raquel Urtasun[1,2]
{james.tu,mren3,manivasagam,ming.liang,byang10,frank.cheng,urtasun}@uber.com
rdu@princeton.edu
[1]Uber ATG [2]University of Toronto [3]Princeton University

Abstract

Modern autonomous driving systems rely heavily on deep learning models to process point cloud sensory data; meanwhile, deep models have been shown to be susceptible to adversarial attacks with visually imperceptible perturbations. Despite the fact that this poses a security concern for the self-driving industry, there has been very little exploration in terms of 3D perception, as most adversarial attacks have only been applied to 2D flat images. In this paper, we address this issue and present a method to generate universal 3D adversarial objects to fool LiDAR detectors. In particular, we demonstrate that placing an adversarial object on the rooftop of any target vehicle to hide the vehicle entirely from LiDAR detectors with a success rate of 80%. We report attack results on a suite of detectors using various input representation of point clouds. We also conduct a pilot study on adversarial defense using data augmentation. This is one step closer towards safer self-driving under unseen conditions from limited training data.

1. Introduction

Modern autonomous driving systems use deep neural networks (DNNs) to process LiDAR point clouds in order to perceive the world [20, 34, 45]. Despite introducing significant performance improvements, DNNs have been previously found to be vulnerable to adversarial attacks when using image inputs [35, 15, 18, 2], where a small perturbation in the input pixels can cause drastic changes in the output predictions. The potential vulnerabilities, in conjunction with the safety-critical nature of self-driving, motivate us to investigate the possibility of disrupting autonomous driving systems with adversarial attacks.

Image perturbations alone however, are not enough for modern autonomous driving systems, which are typically equipped with LiDAR sensors producing point clouds as the primary main sensory input. Several previous works have shown successful attacks [36, 39, 43] with point cloud perturbations, generating salient modifications by adding, removing, and modifying points. Although these attacks work in theory,arbitrary to point clouds are not always physically realizable. For instance, a given point cloud may be impossible to generate from a LiDAR sensor due to the lasers' fixed angular frequencies and light projection geometry.

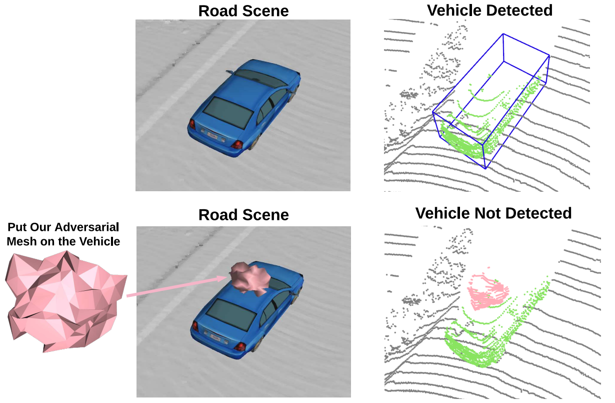


Figure 1: In this work we produce a physically realizable adversarial object that can make vehicles "invisible". After placing the object on the rooftop of a target vehicle, the vehicle will no longer be detected by a LiDAR detector.

Towards generating physically realizable attacks, Cao *et al*. [7] propose to learn an adversarial mesh capable of generating adversarial point clouds with a LiDAR renderer. However, their work only considers learning an adversarial mesh for a few specific frames. As a result, the learned 3D object is not *universal* and may not be reused in other 3D scenes. Moreover, they have only evaluated their attack on a very small in-house dataset that contains around a few hundred frames.

In contrast to [7], we aim to learn a universal and physically realizable adversary. Furthermore, we craft 3D objects in a novel setting where they are placed on top of vehicles as rooftop cargo. Such objects can be used in any scene and on any types of small vehicles, and can hide the entire host vehicle from a strong LiDAR detector [41] with a success rate of 80% at IoU 0.7. By comparison, placing a random object on the rooftop only produces a small drop in detection accuracy. We evaluated our learned adversarial object on a suite of common LiDAR detector architectures that take in various kinds of input representations, and we also report transferability of attacks across these models. Lastly, we conduct a pilot study on adversarial defense using data augmentation and adversarial training. Applying defense mechanisms significantly decreases the likelihood of missing detections of vehicles with strange roof-top cargo, which is a rare-seen but practical scenario for self-driving cars.

The contributions of this paper can be summarized as follows:

1. We propose a universal adversarial attack on LiDAR detectors with physically realizable 3D meshes.
2. We present a novel setting where the adversarial object makes the target vehicle invisible when placed on the vehicle rooftop.
3. We report a thorough evaluation across different detector, each using different input representations.
4. We present a successful defense mechanism via training with data augmentation.

In the following, we first review prior literature on adversarial attacks and in particular point cloud and 3D physical attacks. Next we present details of our proposed method to generate a physically realizable adversarial object, followed by empirical evaluations of our attack and defense on several LiDAR object detectors.

2. Related Work

Despite the impressive performance of deep learning models, they are surprisingly vulnerable to minuscule perturbations. Adversarial attacks add visually imperceptible noise to the input to drastically alters a neural network's output and produce false predictions.

Image attacks: Adversarial examples were first discovered in the context of image classification networks [35]. These vulnerabilities were later discovered in networks performing image detection and semantic segmentation as well [40]. Attacks can either be *white box* [15, 28, 26], where the target model's weights are available, or *black box* [4, 29, 10], where the adversary can only query the outputs of the target model. Various defense mechanisms, including adversarial training [15, 26], denoiser [21], Bayes classifier [33], certified defense [37, 32] have been proposed, and shown effective on a certain range of attack types.

Point cloud attacks: With the rise of LiDAR sensors in robotics applications such as self-driving, point cloud data has become a popular input representation. Some recent research demonstrated the possibility of adversarial attacks on networks that take point cloud data as input. [42] proposed to add, remove, or perturb points; whereas [39] added clusters of adversarial points. [44, 36] proposed saliency-based approaches for removing points. Several structured methods have been also introduced to perturb point clouds with the goal of preserving physical fidelity [23].

Physical world attacks: Perturbing image pixels or point locations alone may not guarantee that the attack can happen in the physical world. To address this issue, several works have produced physical adversaries and expose real world threats. [18] studies whether the image pixel perturbations can be realized by a physical printer. [5] produces a universal and robust adversarial sticker. When placed on any image with any pose, the sticker induces a targeted false classification. [13] proposes to train the attack with different view angles and distances to make it robust. [2] synthesizes robust adversarial 3D objects capable of fooling image classifiers when rendered into an image from any angle. [43, 24] consider perturbing other photo-realistic properties such as shape normal and lighting, using a differentiable renderer. Nevertheless, in these approaches, the final outputs are still projected to 2D image space.

In the context of self-driving and LiDAR perception, [7] propose *LidarAdv*, a method to learn adversarial meshes to fool LiDAR detectors. Our work is different from theirs in several important ways. First, *LidarAdv* only considers one frame during learning and hence is input specific; whereas we train our adversary on all frames and all vehicles, creating a universal adversary. Second, our adversary can be placed on a vehicle roof to hide it, whereas their adversarial object does not interact with other real world objects. Lastly, we are the first to conduct a thorough evaluation of a physically realizable adversarial attack on a suite of detectors and on a public large scale LiDAR benchmark. Besides learning a physical mesh, [6] proposes to use laser devices to spoof LiDAR points. These laser devices, however, are more difficult to set up and it is not trivial to create consistent point clouds as the sensor moves.

3. Physically Realizable Adversarial Examples

In this section, we present our method for learning an adversarial object to attack LiDAR detectors. There are many possible ways to place an object in a scene for an adversarial effect, as explored in [7]. For example, one can hide the inserted adversarial object or change the object label (*e.g.*, making the detector believe that the adversarial object is a pedestrian). In this paper, we instead focus on a novel setting, where we place the object on the rooftop of a vehicle and hide the vehicle from the detector, hence creating an "invisible" car, illustrated in Figure 2. Such a procedure can be easily reproduced in the real world and is also plausible even without the presence of a malicious attacker, since cars occasionally carry pieces of furniture or sports equipment on their rooftops.

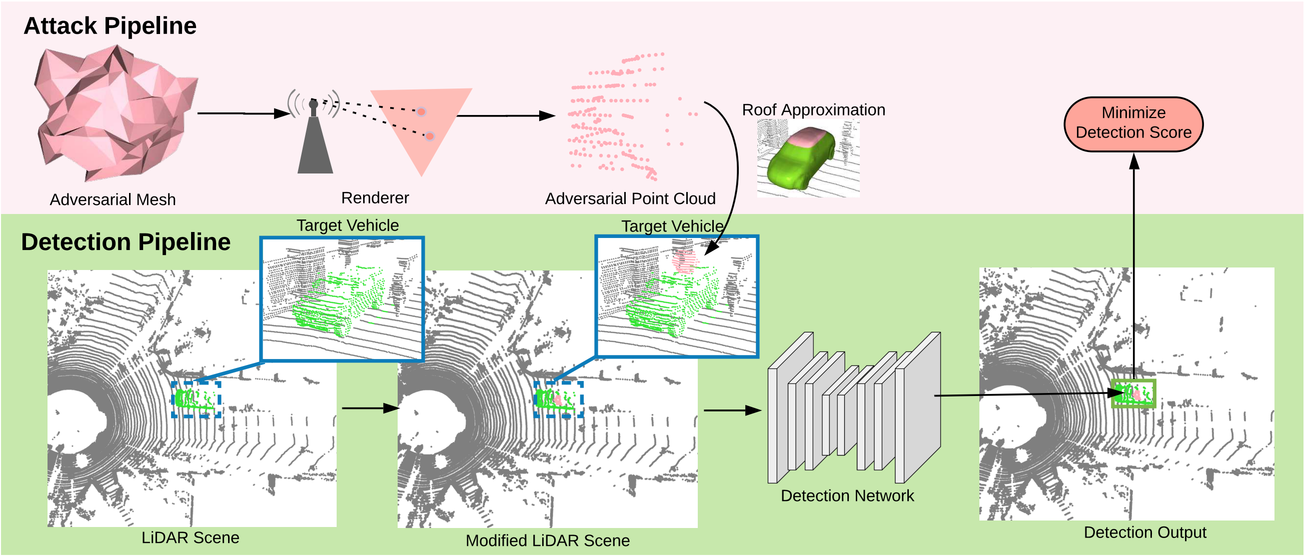


Figure 2: Overall adversarial example generation pipeline. We start from a mesh representation, and use a LiDAR renderer to obtain the point clouds. By using roof approximation techniques, we attach the adversarial point cloud on top of the target vehicle, and we modify the mesh vertices so that the detection confidence score of the target vehicle is minimized.

For the rest of this section, we first describe the 3D representation of the adversarial example and how to render it into a point clouds We then present our adversarial example generation algorithms. Lastly, a simple defense algorithm based on data augmentation is presented.

3.1. Surface Parameterization

Many parameterizations exist for 3D objects, including voxels, meshes, and implicit surfaces [16]. Voxels are easy to compute but require significantly more memory than the alternatives to produce a high level of detail. Implicit surfaces, on the other hand, provide compact representations but are harder to render since they require solving for the numerical roots of the implicit functions. In this paper, we choose to represent our adversary with a mesh since it benefits from compact representations and allows for efficient and precise rendering. Given a mesh, we can compute the exact intersections of rays analytically and in a differentiable manner. The latter is important since it allows us to take gradients efficiently for white box attacks. Furthermore, meshes have previously demonstrated high-fidelity shape generation results on faces and human bodies [3, 22].

During the learning of the adversarial mesh, following prior literature [25, 17], we deform a template mesh by adding local learnable displacement vectors $\Delta \mathbf{v}_i \in \mathbb{R}^3$ for each vertex and a global transformation for the entire mesh,

$$\mathbf{v}_i = \mathbf{R}(\mathbf{v}_i^0 + \Delta \mathbf{v}_i) + \mathbf{t}, \tag{1}$$

where $\mathbf{v}_i^0$ is the initial vertex position, and $\mathbf{R} \in SO(3)$ is a global rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ is a global translation vector. To ensure physical feasibility, box constraints are applied to the mesh vertices as well as the global translation.

In the experiments where we initialize the mesh from an isotropic sphere, $\mathbf{R}$ is fixed to be the identity matrix, since the sphere is rotation invariant. In the experiments where we deform common objects, we constrain $\mathbf{R}$ to be rotations on the x-y plane:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where θ is the learnable rotation angle.

3.2. LiDAR Simulation

We aim to add an adversarial mesh Z into the scene in a realistic manner and choose the roof of vehicles as the location for placement, as placing objects on top is easier due to gravity and does not interfere with adjacent traffic. Furthermore, objects on top of vehicles are less prone to occlusion, whereas areas like the front hood or trunk top may be blocked by another vehicle. Finally, this is a realistic scenario as it common to strap furniture, canoes, bicycles, and other large items on top of vehicles. In this section we first describe how to render a mesh into LiDAR points. Next we introduce a technique to locate the rooftop region from a vehicle point cloud, where we can place the adversary.

LiDAR point rendering: We then use location of the mesh in the scene to sample nearby rays γ with the same angular frequencies as the LiDAR sensor used to generate the original LiDAR scene. Given rays γ and mesh Z, the adversarial points are rendered with a differentiable raycaster R. We compute the intersection of rays and mesh faces with the Moller-Trumbore intersection algorithm [27]. We refer readers to the supplementary materials for more details on this. Then, we take the union of the rendered adversarial points and the original points to create the modified scene.

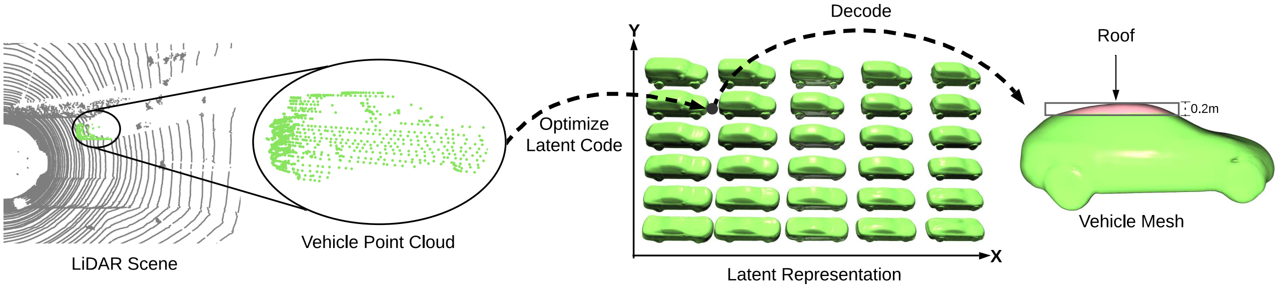


Figure 3: Approximating rooftop from vehicle point clouds. We build a low dimensional representation of our vehicle object bank using PCA, and embed the target vehicle point clouds by optimizing the latent code. The top 0.2m is then cropped to be the rooftop region.

Rooftop fitting: To approximate the center of a vehicle's roof from its point cloud, as illustrated in Figure 3, we first fit a CAD model to the point cloud. Inspired by [12], we represent our internal collection of vehicle models as signed distance functions (SDFs), denoting as $F(\cdot;\theta)$, and project this library of vehicles into a latent space using PCA. Since SDFs implicitly represent 3D surfaces as its zero level-set, we optimize the latent code z such that all ground truth vehicle points evaluate as close to 0 as possible. Concretely, given a vehicle bounding box (x, y, w, h, α), and a set of points $P = \{p : p \in \mathbb{R}^3\}$ within the box, we find the optimal latent code z^* such that

$$z^* = \arg\min_z \sum_{p \in P} F(p; \theta(z))^2. \tag{3}$$

We then apply marching cubes [16] on $F(\cdot;\theta(z^*))$ to obtain a fitted CAD model. Lastly, we use vertices within the top 0.2m vertical range of the CAD model to approximate the roof region.

During the attack, we place the adversarial mesh Z with a fixed pose relative to the roof center of a target vehicle. Given a vehicle bounding box (x, y, w, h, α), we compute the roof center (r_x, r_y, r_z) and apply transformation matrix

$$T = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & r_x \\ \sin\alpha & \cos\alpha & 0 & r_y \\ 0 & 0 & 1 & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

on the adversarial object.

3.3. Adversarial Example Generation

In this section, we first introduce the objective function being optimized to learn adversarial examples. Both white box and black box attack algorithms are presented next.

3.3.1 Objective

The overall loss function is a combination of the adversarial loss and the Laplacian loss for mesh smoothness:

$$\mathcal{L} = \mathcal{L}_{\text{adv}} + \lambda \mathcal{L}_{\text{lap}}. \tag{5}$$

To generate an adversarial example, we search for vertex perturbation $\mathbf{v}$ and global transformation parameters $(\mathbf{R}, \mathbf{t})$ that minimize the loss function.

For the adversarial loss, following prior work [40], we also find it necessary to suppress all relevant bounding box proposals. A proposal is relevant if 1) its confidence score is greater than 0.1 and 2) if its IoU with the ground truth bounding box is also greater than 0.1.

Our adversarial objective minimizes the confidence of the relevant candidates:

$$\mathcal{L}_{\text{adv}} = \sum_{y,s \in \mathcal{Y}} -\text{IoU}(y^*, y) \log(1 - s), \tag{6}$$

where $\mathcal{Y}$ is the set of relevant bounding box proposals and each proposal y has a confidence score s. We use binary cross entropy to minimize the confidence score of the relevant proposals, weighed by the IoU with the ground truth bounding box y^*. Here, we choose to target negative classification labels instead of other bounding box parameters because missing detections are the most problematic.

In addition, a Laplacian loss [25] is applied to regularize mesh geometry and maintain surface smoothness:

$$\mathcal{L}_{\text{lap}} = \sum_i \|\delta_i\|_2^2, \tag{7}$$

where δ_i is the distance from v_i to the centroid of its immediate neighbors $N(i)$:

$$\delta_i = v_i - \frac{1}{\|N(i)\|} \sum_{j \in N(i)} v_j. \tag{8}$$

3.3.2 Attack Algorithms

In this section we provide details for both white box and black box attacks for learning mesh vertices.

White box attack: In a white box setting, we simulate the addition of the adversary in a differentiable manner, and hence can take the gradient from the objective $\mathcal{L}$ to the mesh vertices. In addition, we re-parameterize local and global displacement vectors to apply box constraints, as [8] have demonstrated issues with other alternatives. Specifically, since clipping parameters during *projected gradient descent* creates a disparity between parameter updates and momentum updates, we instead re-parameterize mesh vertices to inherently obey box constraints:

$$\mathbf{v}_i = \mathbf{R}(\mathbf{b} \odot \mathrm{sign}(\tilde{\mathbf{v}}_i^0) \odot \sigma(|\tilde{\mathbf{v}}_i^0| + \Delta\tilde{\mathbf{v}}_i)) + \mathbf{c} \odot \tanh(\tilde{\mathbf{t}}),$$

where $\odot$ denotes element-wise multiplication, σ denotes the sigmoid function, $\mathbf{b} \in \mathbb{R}^3$ define limits on size, and $\mathbf{c} \in \mathbb{R}^3$ define limits on translation. $\sigma(\tilde{\mathbf{v}}_i^0) = \mathbf{v}_i^0/\mathbf{b}$ is the normalized initial position of vertex and $\tanh(\tilde{t}) = \mathbf{t}/\mathbf{c}$ is the normalized global translation. The sign function constrains each vertex to stay in its initial quadrant.

Black box attack: A gradient-based attack is not always feasible in point cloud perception due to non-differentiable preprocessing stages that are common in modern point cloud detection models [41, 19]. For example, models like PIXOR [41] represent the input as occupancy voxels, preventing gradients from reaching the point cloud. To address this problem, we employ a genetic algorithm [1] to update the mesh parameters. Here, a population of candidates meshes are evolved to maximize the fitness score $-\mathcal{L}$. At every iteration, the candidate with the highest fitness is preserved while the rest are replaced. New candidates are generated by sampling mesh parameters from a pair of old candidates, with sampling probability proportional to fitness score. We then add gaussian noise to some new candidates sampled with a mutation probability. To jointly optimize over all samples, we perform inference on multiple examples and take the average fitness score at each iteration. In this black box setting, we find re-parameterization unnecessary for gradient-free optimization.

3.4. Defense Mechanisms

Given that rooftop objects are rarely observed in the training distribution and that our attack produces examples that are heavily out-of-distribution, we first propose random data augmentation as a simple defense mechanism. Next, we consider adversarial training [8] for a stronger defense against adversarial attacks.

Data augmentation: When training with data augmentation, in every frame we generate a random watertight mesh and place it on a random vehicle using the methods presented previously. This method is not specific to the type of optimization employed by the attacker (*e.g.* white box or black box) and hence may generalize better when compared to regular adversarial training [15].

To generate a random watertight mesh, we first sample a set of N vertices $V \in \mathbb{R}^{N\times 3}$ from a Gaussian $\mathcal{N}(0, \sigma)$ and apply incremental triangulation to obtain a set of connected tetrahedrons Q. We then stochastically remove M boundary tetrahedrons that do not disconnect Q into separate components. Finally, we take the remaining boundary faces of Q to obtain a watertight surface.

Adversarial training: While adversarial training has empirically been found to be robust, it is expensive and infeasible when the cost of training an adversary is high. Thus, we employ a method similar to [38] and take one mesh update step per model update instead of a full optimization cycle. During training, the adversary is randomly re-initialized every k steps using the same mesh generation method.

4. Experiments

In this section, we first discuss the datasets and models used in our experiments in Section 4.1 and 4.2, and the experimental setup in Section 4.3. We then present experimental results on 1) white box and black box attacks, 2) attacks and transferability on various detection backbones, 3) common object attacks, and 4) adversarial defense using data augmentation training.

4.1. Datasets

We use the KITTI dataset [14] for training and evaluation of our attacks. KITTI contains LiDAR point clouds and 3D bounding box labels for objects seen by the front camera of the autonomous vehicle. For our experiments, we focus on the "Car" class only and consider each object in a scene as a separate sample. Since our method relies on fitting meshes to point clouds, we discard all samples with less than 10 points. This results in 6864 vehicles in the training set and 6544 vehicles in the validation set. We do not use the test set as labels are not publicly available. For evaluation we only consider bounding boxes from a bird's eye view.

4.2. Target LIDAR Detector Models

In this work, we attack detector models that process point clouds exclusively without any auxiliary inputs, since we only learn the shape of the adversarial mesh. We cover a variety of detector architectures with different input representations of the point cloud. Specifically we consider the following models:

- **PIXOR** [41] is a detection network that processes input point clouds into occupancy voxels and generates bounding boxes in a bird's eye view.
- **PIXOR (density)** is a variant of PIXOR using density voxels as inputs. The value of each voxel is calculated from bilinear interpolation of nearby points' distance to the voxel center: $\sum_p (1-|p_x - v_x|)(1-|p_y - v_y|)(1-|p_z - v_z|)$. The density variant allows us to compute gradients to point clouds easier.
- **PointRCNN** [34] does not voxelize and instead processes the raw point cloud directly using a PointNet++ [31] backbone.
- **PointPillar** [19] groups input points into discrete bins from BEV and uses PointNet [30] to extract features for each pillar.

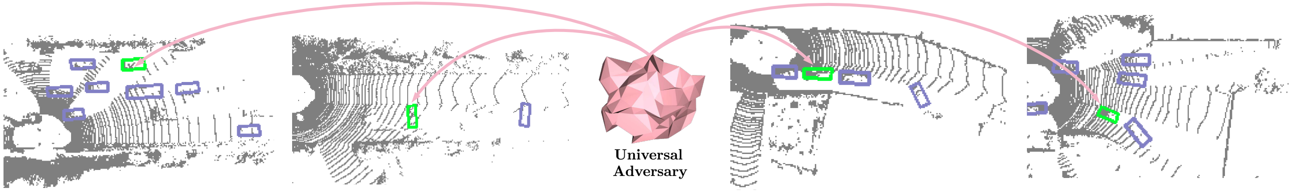


Figure 4: Visualization of our universal adversarial object hiding different car instances at various orientations and locations.

Source \ Target	PIXOR-1	PIXOR-2
PIXOR-1	**77.3%**	53.9%
PIXOR-2	73.5%	**72.1%**

Table 1: Attack transferability between two PIXOR models with different seeds.

Source \ Target	PIXOR	PIXOR (d)	PointRCNN	PointPillar
PIXOR [41]	**77.3%**	66.0%	20.1%	8.2%
PIXOR (density) [41]	66.4%	**80.9%**	20.0%	7.7%
PointRCNN [34]	33.3%	33.7%	**32.3%**	20.5%
PointPillar [19]	54.9%	38.4%	28.4%	**57.5%**

Table 2: Attack transferability among different detector models.

Since we limit to the scope of learning the mesh shape only, we use the version of the above detectors that do not take LiDAR intensity as input.

4.3. Experimental Setup

Implementation details: In our experiments, we initialize the adversarial mesh to be a unit isotropic sphere with 162 vertices and 320 faces and scale it by $\mathbf{b} = (b_x, b_y, b_z)$ = (0.7m, 0.7m, 0.5m). Maximum global offset is 0.1m on the x, y direction and no offset is allowed on the z direction to prevent the mesh from moving into the vehicle or hovering in mid air. For the Laplacian loss, we set $\lambda = 0.001$. During simulation, rays are sampled according to specs of the Velodyne HDL-64E sensor used to generate the datasets.

For the gradient-based optimization in white box attacks, we use Adam with learning rate 0.005. For the genetic algorithm in black box attacks, we initialize mutation std at 0.05, mutation probability at 0.01, use a population size of 16, and average 100 queries to compute fitness. We decay the mutation std and probability by a factor of 0.5 if the running fitness has not improved in 100 generations.

4.4. Evaluation Metrics

We consider the following two metrics for attack quality:

- **Attack success rate:** Attack success rate measures the percentage at which the target vehicle is successfully detected originally and but not detected after the attack. We consider a vehicle successfully detected if the output IoU is greater than 0.7.
- **Recall-IoU curve:** Since attack success rate depends on the IoU threshold, we also plot the recall percentage at a range of IoU threshold to get a more thorough measure of attack quality.

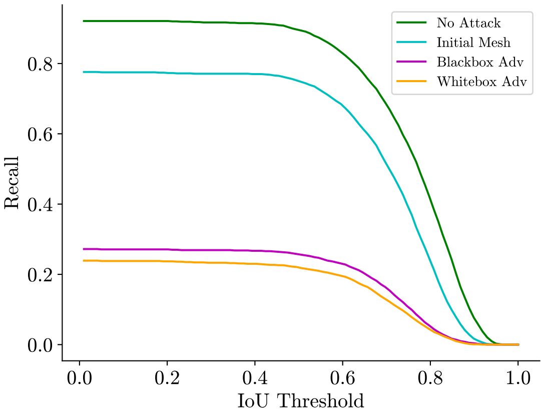


Figure 5: Visualization of detection recall across range of IoUs for PIXOR with density voxels. Difference between white box and black box attacks is very small. Initial mesh does not affect detection as much as adversarial mesh.

4.5. Results and Discussion

Comparison of White Box and Black Box We conduct a white box and black box attack on a variant of PIXOR with density voxels. We visualize the IoU-recall curve for both experiments in Figure 5. and show that the attacks significantly drop the recall. When we use the initial icosphere mesh as a baseline, it has little impact on detection even though it is of the same size as the adversary. We further compare our black box attack against the white box alternative and show that they achieve similar performance.

Transferability Across Identical Architectures We investigate the transferability of adversarial examples across similar models. To this end, we train two variations of the original PIXOR model using different seeds and learn adversarial meshes for each model separately. We then evaluate transferability between the pair of models using attack success rate as the metric. Results are summarized in Table 1 and there is a high degree of transferability between models with identical architecture. This allows strong transfer attacks with only knowledge of model architecture. [29].

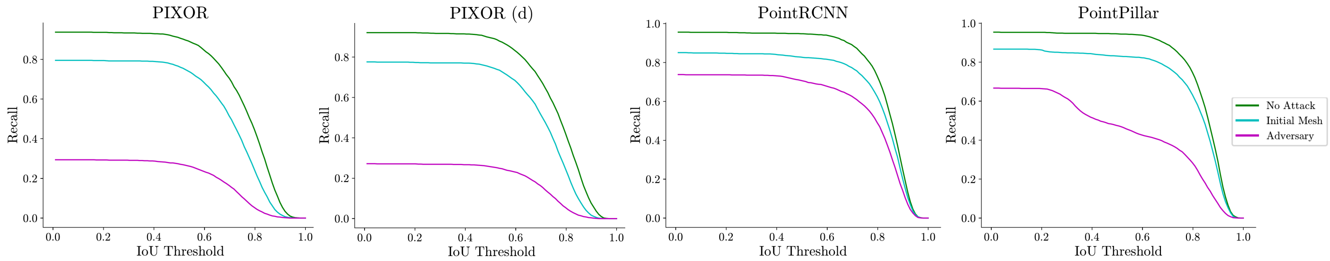


Figure 6: IoU-Recall curve for black box attacks on different detector architectures.

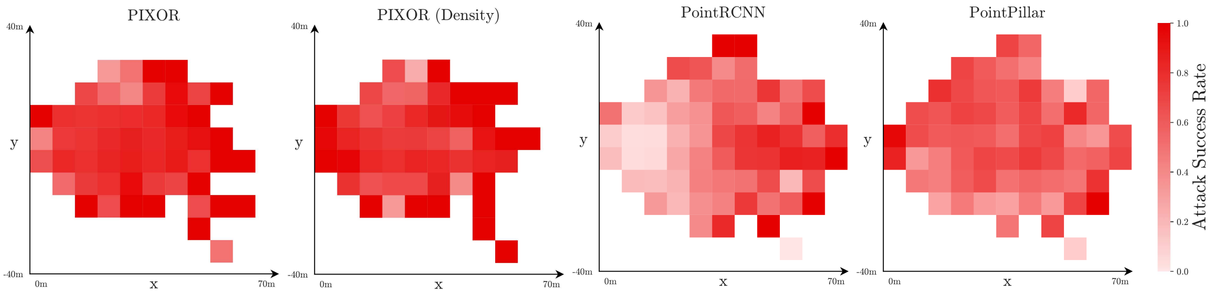


Figure 7: Bird's eye view visualization of attack success rate at various locations in the scene using different detector models.

Transferability Across Input Representations In this section we attack four models described in Section 4.2, and use only black box attacks due to non-differentiable layers in some models. We provide IoU-Recall curves in Figure 6. Again, transferability between the models is considered and results are shown in Table 2.

First, our attack is able to hide objects with high probability on the PIXOR models and PointPillar but is significantly weaker on PointRCNN. We hypothesize that this is because PointRCNN treats every point as a bounding box anchor. Therefore, vehicles close to the sensor register significantly more LiDAR points and proposals, making it extremely difficult to suppress all proposals. We verify this hypothesis by visualizing the attack success rate at various locations across the scene in Figure 7. The success rate on attacking PointRCNN is close to 0 near the LiDAR sensor but grows substantially higher as the distance to the sensor increases and the number of points decreases.

In terms of transferability, the occupancy and density variants of PIXOR can share a significant portion of adversarial examples. The attacks generated from PointPillar and PointRCNN can also be used to attack PIXOR, but not vice versa. This suggests that additional layers of point-level reasoning before aggregating on the z-dimension probably make the model more robust to rooftop objects.

In addition, two variations of PIXOR have different vulnerable regions even though they share the same backbone. Specifically, we note that vehicles close to the LiDAR sensor are the easiest targets when using a density voxel input representation. In contrast, vehicles closeby are the most robust to attacks when using occupancy voxels. We speculate that this is an effect of the input precision. For density voxels, the number of points is significantly higher near the LiDAR sensor, contributing to higher input precision, whereas occupancy voxels can only show a binary indicator whether a voxel contains a point or not.

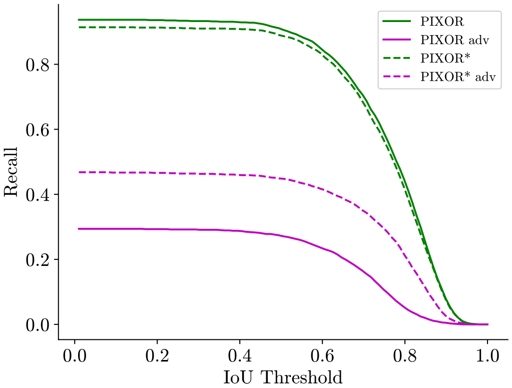


Figure 8: We perform our attack on two similar models. PIXOR converts the input point cloud to occupancy voxels and PIXOR* separates points into columns and extracts features with a pointnet backbone. Although the models have the same backbone architecture, PIXOR* is significantly more robust due to the input representation.

Based on the above observations, we conclude that the choice of input representation and detection scheme may have significant implications on robustness to adversarial examples. For a more concrete comparison, we consider a variant of PIXOR using PointPillar's pillar representation instead of voxelization and keep the backbone architecture identical. We compare this variant, PIXOR* against PIXOR and show the results in Figure 8. Here, we can see that

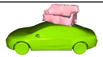
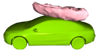
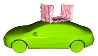
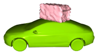
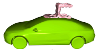
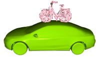
Object	Initial	Success	Classification	Adversarial	Success	Classification	Dimensions
Couch		23.3%	93.1 %		**68.6%**	93.8 %	1.69m x 0.85m x 0.94m
Canoe		26.9%	99.6 %		**59.5%**	99.9 %	3.51m x 0.81m x 0.68m
Table		18.9%	99.8 %		**48.0%**	99.7 %	1.57m x 0.83m x 0.86m
Cabinet		20.7%	93.4 %		**54.1%**	94.2 %	1.29m x 0.91m x 0.76m
Chair		14.1%	99.9 %		**23.3%**	99.9 %	1.42m x 0.64m x 0.71m
Bike		19.6%	94.4 %		**32.4%**	92.3 %	1.70m x 0.76m x 1.08m

Table 3: Adversaries resembling common objects that could appear on the rooftop. Attack success rates on the initial and adversarial configurations are shown. A ShapeNet classifier stably recognizes our adversaries as the correct object class.

	Arbitrary	Couch	Canoe	Table	Cabinet	Chair	Bike	AP
Original	77.3%	68.6%	59.5%	48.0%	54.1%	23.3%	32.4%	74.37
Augmentation	14.4%	12.4%	19.6%	11.5%	7.3%	6.6%	14.0%	**74.92**
Adv Train	**10.8%**	**5.6%**	**18.7%**	**4.9%**	**5.3%**	**6.3%**	**11.4%**	73.97

Table 4: Attack success rates before and after applying defense training on PIXOR. In the final column, we evaluate the models on the standard KITTI validation set and show the average precision (AP) at 0.7 IoU.

with even with identical backbones and training routines, PIXOR* is significantly more robust to our attack purely due to a different input representation.

Common Objects In this section, to make attacks more realistic, we learn adversaries that resemble common objects that may appear on top of a vehicle in the real world. Instead of deforming an icosphere, we initialize from a common object mesh and deform the vertices while constraining the maximum perturbation distances. We choose couch, chair, table, bike, and canoe as six common object classes, and we take the object meshes from ShapeNet [9]. We apply uniform mesh re-sampling in meshlab [11] to reduce the number of faces and produce regular geometry prior to deformation. In these experiments we limit the maximum vertex perturbation to 0.03m so that the adversary will resemble the common object, and limit translation to 0.1m, and allow free rotation. In Table 3, we present the visualizations, results, and dimensions of the common objects. Moreover, the identity of the adversarial objects are unambiguous to a human, and we also verify that a PointNet [31] classifier trained on ShapeNet [9] is also able to correctly classify our perturbed objects. This confirms the possibility that the placement of common objects can also hurt LiDAR detectors.

Adversarial Defense We employ our proposed defense methods by retraining a PIXOR model with random data augmentation and adversarial training. To generate random meshes for data augmentation, we uniformly sample N from $[50, 200]$, M from $[0, 300]$, and we sample vertices from Gaussian $\mathcal{N}(0, 0.5)$. If all tetrahedrons are removed by decimation, the sampling process restarts. During training, for every scene we sample one vehicle at random for data augmentation. We only augment vehicles with at least 10 points in the scene, otherwise it is too difficult to fit a mesh for roof approximation. During adversarial training, we set $k = 30$ and alternate between updating the mesh and the model.

For evaluation, we re-train an adversarial mesh on the defended model and observe that the attack success rate is reduced significantly, as shown in Table 4. In addition, we also launch attacks with common objects on the defended model and observe similar findings. Furthermore, for the standard detection task, our defended models achieve similar or better performance when evaluated on the KITTI validation set. Nevertheless, the defense is not yet perfect since there is still 5-20% attack success rate remaining. With more computational resources, full adversarial training could possibly close this gap.

5. Conclusion

We propose a robust, universal, and physical realizable adversarial example capable of hiding vehicles from LiDAR detectors. An attacker can 3D print the mesh and place it on any vehicle to make it “invisible” without prior knowledge of the scene. The attack will consistently cause target vehicles to disappear, severely impeding downstream tasks in autonomous driving systems. Even without any malicious intent, we show that problematic shapes can coincidentally appear with common objects such as a sofa. We further show that training with data augmentation using random meshes can significantly improve the robustness, but unfortunately still not 100% secure against our attack. By demonstrating the vulnerability of LiDAR perception against universal 3D adversarial objects, we emphasize the need for more robust models in safety-critical robotics applications like self-driving.

References

[1] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1111–1119, 2019. 5

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. 2018. 1, 2

[3] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. In *CVPR*, 2018. 3

[4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018. 2

[5] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[6] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *CCS*, 2019. 2

[7] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418*, 2019. 1, 2

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *SP*, 2017. 4, 5

[9] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 8

[10] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec@CCS*, 2017. 2

[11] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 8

[12] Francis Engelmann, Jörg Stückler, and Bastian Leibe. SAMP: shape and motion priors for 4d vehicle reconstruction. In *WACV*, pages 400–408, 2017. 4

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *CVPR*, 2018. 2

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015. 1, 2, 5

[16] John F. Hughes, Andries Van Dam, Morgan McGuire, DAvid F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996. 3, 4

[17] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018. 3

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017. 1, 2

[19] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 5, 6

[20] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 1

[21] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 2018. 2

[22] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *CVPR*, 2018. 3

[23] Daniel Liu, Ronald Yu, and Hao Su. Adversarial point perturbations on 3d objects. *arXiv preprint arXiv:1908.06062*, 2019. 2

[24] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *ICLR*, 2019. 2

[25] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. 2019. 3, 4

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 2

[27] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *J. Graph. Tools*, 2(1):21–28, Oct. 1997. 3

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, 2016. 2

[29] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *AsiaCCS*, 2017. 2, 6

[30] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 5

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 5, 8

[32] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *ICLR*, 2018. 2

[33] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *ICLR*, 2019. 2

[34] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 5, 6

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014. 1, 2

[36] Matthew Wicker and Marta Kwiatkowska. Robustness of 3d deep learning in an adversarial setting. In *CVPR*, 2019. 1, 2

[37] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018. 2

[38] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. 5

[39] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3d adversarial point clouds. In *CVPR*, 2019. 1, 2

[40] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *ICCV*, 2017. 2, 4

[41] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 1, 5, 6

[42] Jiancheng Yang, Qiang Zhang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. Adversarial attack and defense on point sets. *arXiv preprint arXiv:1902.10899*, 2019. 2

[43] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L. Yuille. Adversarial attacks beyond the image space. In *CVPR*, 2019. 1, 2

[44] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *ICCV*, 2019. 2

[45] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1