SHIFT3D: Synthesizing Hard Inputs For Tricking 3D Detectors

Hongge Chen*[1], Zhao Chen[1], Gregory P. Meyer[1], Dennis Park[1],
Carl Vondrick[1], Ashish Shrivastava[1], and Yuning Chai†[1]

[1]Cruise LLC

Abstract

We present SHIFT3D, a differentiable pipeline for generating 3D shapes that are structurally plausible yet challenging to 3D object detectors. In safety-critical applications like autonomous driving, discovering such novel challenging objects can offer insight into unknown vulnerabilities of 3D detectors. By representing objects with a signed distanced function (SDF), we show that gradient error signals allow us to smoothly deform the shape or pose of a 3D object in order to confuse a downstream 3D detector. Importantly, the objects generated by SHIFT3D physically differ from the baseline object yet retain a semantically recognizable shape. Our approach provides interpretable failure modes for modern 3D object detectors, and can aid in preemptive discovery of potential safety risks within 3D perception systems before these risks become critical failures.

1. Introduction

As 3D computer vision models become ubiquitous in real-world applications, their reliability becomes a critical safety concern should they fail in undetected or unaddressed ways. Autonomous vehicles, for example, rely heavily on 3D vision, and failures on the road within these systems can lead to collisions and other dangerous events.

Like most statistical models, contemporary 3D detectors are typically susceptible to failure on rare or unknown events encountered in the real world. Most solutions in the field tend to be *reactive*, with more data being collected a posteriori [6, 10, 25] or models being ad-hoc retrained to target specific already-labeled data [5, 16]. However, when a single failure can lead to loss of life on the road, there is an urgent need for detecting these failures in a *proactive* way. Therein lies one of the central challenges of deploying safe AI in practice: a model trained on a finite dataset can perform poorly on data it has not seen, but by definition we cannot just collect unknown data to better understand what a model does not know.

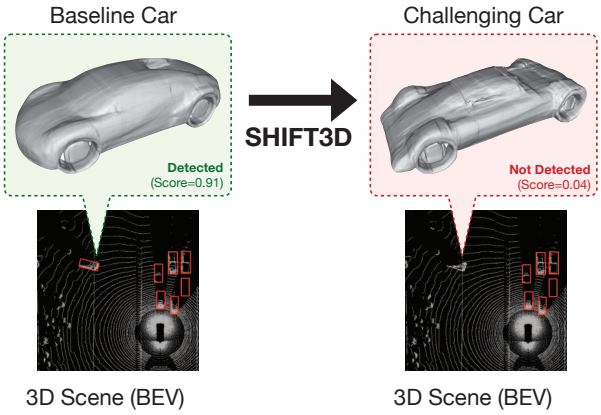


Figure 1: SHIFT3D creates challenging 3D objects from a baseline real object and inserts them with realistic occlusions into a point cloud scene. We also synthesize occlusion patterns, therefore the synthetic objects can be placed in various locations in the scene.

In this paper, we propose SHIFT3D (**S**ynthesizing **H**ard **I**nputs **f**or **T**ricking **3**D **D**etectors), an approach to proactively synthesizing hard examples for 3D object detection through generative modeling. SHIFT3D works by perturbing pre-existing objects into hard examples for 3D vision models, and as we demonstrate, works well in the autonomous driving setting. The perturbative approach provides multiple benefits for us: we can explore new parts of the input distribution which are not explicitly represented by the training data, and the perturbation in the shape space allows us to interpret precisely what changes in shape are most salient in making the example challenging. Since interpretability is a key benefit of this process, it is also important for these perturbations to be *naturalistic*, with changes easily recognizable by the human eye. We use *naturalistic*

*hongge.chen@getcruise.com
†yuning.chai@getcruise.com

primarily to contrast with the traditional *adversarial* class of perturbations [21, 29, 9, 17], where minuscule changes produce data that challenges the deep model, but which are imperceptible to the human eye and thus not very helpful for diagnosing model scalability issues in the real world.

Like traditional methods of adversarial attack [9], we use gradients as our primary signal to generate challenging input deformations which then confuse the downstream model. However, unlike traditional methods of adversarial attack, our method creates 3D objects with observable changes within the object topology that reflect meaningful challenges to the downstream model. To ensure this property, we perform gradient perturbations not on the 3D object directly but only in an appropriate latent space, and allow a decoder to interpret the perturbed latent space into a final 3D object. In this work, the decoder is a pre-trained signed distance function (SDF) network [18], which is a popular model to decode a latent vector into a 3D surface. Additionally, we can also perturb the pose of the object without shape deformation. Our synthesized examples are also robust to background scene choice and to insertion location, making it easy to turn a challenging SHIFT3D object into a fully realized challenging scene.

SHIFT3D operates as follows: a baseline object is taken from some reference mesh and we obtain a latent encoding $\boldsymbol{z}$ for the baseline object from a pretrained DeepSDF [18] model. The DeepSDF object is then placed in a new point cloud at pose $\boldsymbol{\theta}$ (position and orientation), with postprocessing to add occlusions so the insertion will look physically plausible. This new scene is sent through a pretrained 3D object detector, which assigns a detection score to the inserted object. This pipeline allows us to differentiate the detection score all the way back to the DeepSDF shape encoding $\boldsymbol{z}$, along with the pose parameters $\boldsymbol{\theta}$. We then find the gradients that *maximize* the detection error and apply these perturbations to the latent encoding as well as to $\boldsymbol{\theta}$, which we then decode into a final 3D object. Whether we change $\boldsymbol{z}$ (shape) or $\boldsymbol{\theta}$ (pose), we will demonstrate that these final 3D objects have reliably low detection scores and visually look substantially different from the baseline object.

Our main contributions within this work are as follows:

- We introduce SHIFT3D, a pipeline to generate challenging 3D objects and insert them with realistic occlusions into a point cloud scene.
- We show that SHIFT3D can be made fully differentiable through an implicit differentiation setup, which can extend perturbations beyond the shape of the 3D object and to its placement geometry as well.
- We test SHIFT3D in the autonomous driving setting, and show that the objects generated by SHIFT3D consistently mislead our LiDAR-based 3D detector and are robustly transferable between different locations within a scene and different scenes as well.

2. Related Works

Perception is a critical aspect of ensuring safety and security in autonomous driving systems. As the use of LiDAR sensors increases in autonomous vehicles, point cloud data has become a common input representation. However, recent research has shown that adversarial attacks are effective against point cloud models as well. Various techniques have been proposed to perturb, add, or remove points from the point cloud data [26, 30, 24, 15, 14, 32, 12].

3D adversarial attacks on autonomous vehicle systems usually focus on more structurally natural or physically realizable perturbations. LidarAdv [3] proposed creating meshes that can deceive LiDAR detectors by using mesh vertices to generate adversarial objects, which were 3D-printed. Similar techniques were employed in [22], where mesh objects were rendered on top of vehicles in LiDAR log data, which enabled the vehicle to evade detection. Going a step further, [2] created adversarial objects that could deceive both cameras and LiDAR in a multi-sensor fusion-based perception system. Unlike our proposed method, all the studies thus far on generating adversarial 3D objects for autonomous vehicles produce objects without any semantic meaning, or use mesh-based perturbations on existing real objects, creating traditional adversarial examples that are unrecognizable by the naked eye.

There has also been lines of work investigating adversarial attacks by leveraging generative models [11, 33, 19, 27], and generating adversarial images from scratch using rendering engines [31, 23, 28, 1]. However, these works are mostly in image space. To the best of our knowledge, SHIFT3D is the first work that can generate plausible-looking challenging objects and realistically render them into a full 3D point cloud scene.

3. Method

SHIFT3D manipulates the shape and pose of an object in a way that makes it difficult to detect for a 3D vision model, and thus we must create a pipeline that can take an object's shape and pose and render it in a given scene. Importantly, this pipeline will be differentiable with respect to both the shape and pose parameters, which allows us to differentiate the detector loss all the way back to these parameters. These gradients allow perturbations in these parameters that then generate adversarial examples.

To ensure that the pipeline is differentiable, we start with a Sign Distance Function (SDF) [18] network g. We can use g to find a latent shape embedding $\boldsymbol{z}$ that decodes to a given shape, and then insert that shape with pose $\boldsymbol{\theta}$ into a scene with realistically rendered occlusions. The latter is possible because g allows us to query every pre-existing

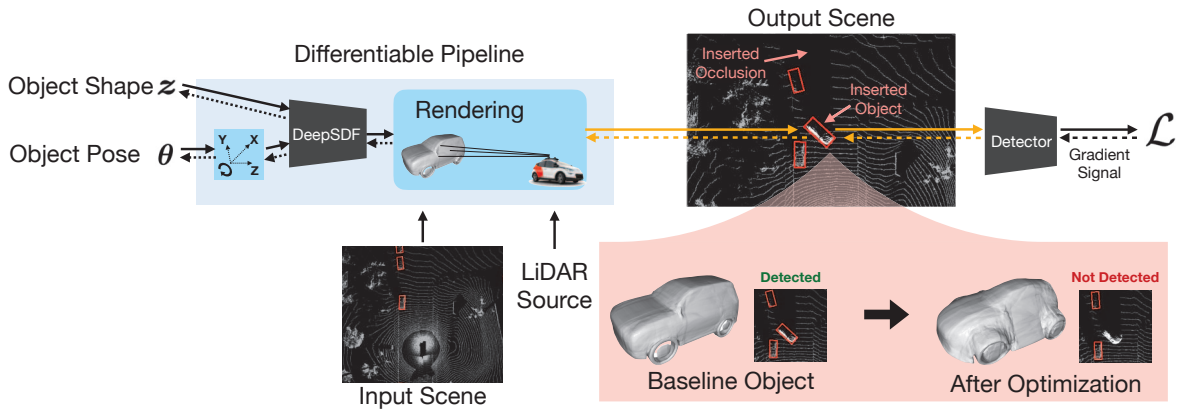


Figure 2: Overview of insertion process of SHIFT3D objects into a LiDAR scene. We use a DeepSDF model to locate points on an object's surface given its shape ($\boldsymbol{z}$) and pose ($\boldsymbol{\theta}$). We reconstruct beams by connecting input sensor position to existing LiDAR points in the input scene. If a beam intersects with the SHIFT3D object, the corresponding point is moved to the object's surface. SHIFT3D objects are optimized to deceive a detector model. Starting with an baseline shape or pose, we render the object in the input scene and then minimize detection model's score. We alternate between rendering the object and minimizing the score for several steps until an adversarial object is produced.

point in the original scene and determine whether that point is now behind the inserted object (i.e. occluded). Occluded points are moved to the inserted object's surface.

We then minimize the detection score of the object detector, f. In this work, we assume we are in the white-box setting where we have access to the detector model weights, and thus can directly differentiate through the detector. Unlike traditional adversarial algorithms that directly perturb the 3D points of the object, we only modify the shape latent parameters, $\boldsymbol{z}$, or the pose parameters, $\boldsymbol{\theta}$; this process helps ensure that the perturbations look natural.

Next, we describe the SDF model, rendering, and SHIFT3D optimization in more detail.

3.1. Object Representation using DeepSDF Model

SDF is an implicit representation of the object's shape and is learned using a deep neural network [18] and a dataset of 3D objects. The DeepSDF function, g, is trained to evaluate to zero for the points on the object's surface. That is, given a shape vector $\boldsymbol{z} \in \mathbf{R}^{d_z}$ of an object and a point $\boldsymbol{x} \in \mathbf{R}^3$ in space, the function $g(\boldsymbol{z}, \boldsymbol{x}) = 0$ if $\boldsymbol{x}$ is on the object's surface, $g(\boldsymbol{z}, \boldsymbol{x}) < 0$ if $\boldsymbol{x}$ is inside the object, and $g(\boldsymbol{z}, \boldsymbol{x}) > 0$ if $\boldsymbol{x}$ is outside the object. In other words, an object's surface in 3D space is precisely the level set of all points $\boldsymbol{x}$ for a given object shape $\boldsymbol{z}$ where $g(\boldsymbol{z}, \boldsymbol{x}) = 0$.

3.2. Object Rendering in LiDAR scene

To render an object with a given pose into a LiDAR scene, we first extend LiDAR beams from the source to points in the original point cloud. If any beam intersects the inserted object, that point is now behind the object, and so we move the corresponding point to the surface of the object. We assume that no object points are dropped due to sensor noise or object material type. See Figure 2 for an overview of the rendering process.

Assume an object has shape $\boldsymbol{z}$ and pose $\boldsymbol{\theta}$. Let $\hat{\boldsymbol{x}}_i$ be a LiDAR point in the input scene and and $\boldsymbol{x}_i$ be the corresponding point in the output scene after rendering the object. We can reconstruct the laser beams in the scene by drawing a straight line connecting the sensor with each $\hat{\boldsymbol{x}}_i$. Let $\hat{\boldsymbol{x}}_i^{(j)}$ be the j^{th} sample point on the line. To determine whether a sampled point, $\hat{\boldsymbol{x}}_i^{(j)}$, lies inside the object, we evaluate it using the SDF function g. Since the SDF function is trained in the object coordinate frame, we need to transfer the point $\hat{\boldsymbol{x}}_i^{(j)}$ into the object coordinate frame using the pose of the object. Let this transformation from sensor coordinate to object coordinate be represented by $T(\cdot; \boldsymbol{\theta})$, which is parameterized by the pose of the object or the relative transform between the sensor and the object. To put it concisely, the point, $\hat{\boldsymbol{x}}_i$, is occluded by the inserted object if $g(\boldsymbol{z}, T(\hat{\boldsymbol{x}}_i^{(j)}; \boldsymbol{\theta})) < 0$ for any j. If this condition is met, then we need to move $\hat{\boldsymbol{x}}_i$ to the object's surface. To do this, we perform a binary search to find the point $\boldsymbol{x}_i$, that lies between $\hat{\boldsymbol{x}}_i^{(j)}$ and the LiDAR sensor, and is on the object's surface within some tolerance ϵ; i.e. , $|g(\boldsymbol{z}, T(\boldsymbol{x}_i; \boldsymbol{\theta}))| < \epsilon$. On the other hand, if $g(\boldsymbol{z}, T(\hat{\boldsymbol{x}}_i^{(j)}; \boldsymbol{\theta})) > 0$ holds for all j, $\hat{\boldsymbol{x}}_i$ remains in its original position, i.e. $\boldsymbol{x}_i \leftarrow \hat{\boldsymbol{x}}_i$. Note that we only use laser beams reconstructed from existing LiDAR points in the input scene, and do not create new laser beams. More etailed information of rendering are presented in Section A of the supplementary material.

3.2.1 Realism Constraints on Object Pose

We can also improve rendering quality by ensuring that the rendered object satisfies the following physical constraints:

The SDF object does not overlap with the scene objects. To prevent the SDF object from overlapping with existing objects in the input scene, we avoid object positions that would result in any point in the scene having an SDF value less than $-\epsilon_{\text{overlap}} < 0$.

The SDF object touches the ground. To ensure that the SDF object is in contact with the ground, we enforce that at least one point in the original scene has SDF value greater than, $\epsilon_{\text{float}} > 0$.

3.3. Natural Adversarial Perturbation of the Object

The rendered scene can be represented with a set of n 3D points $\mathcal{X} = \{\boldsymbol{x}_i\}, i = 1, \ldots, n$. The detector takes $\mathcal{X}$ as input and produces a set of bounding boxes and their corresponding detection scores. Let the set of proposed bounding boxes and their scores be denoted by $\mathcal{Y} = \{(\boldsymbol{y}_j, p_j)\}$, where $\boldsymbol{y}_j$ denotes the bounding box coordinates for the j-th bounding box, p_j is the corresponding detection score. The number of bounding boxes proposed depends on the scene size and the anchor resolution in the detector architecture. We then render the object in the scene with a given $\boldsymbol{z}$ and $\boldsymbol{\theta}$, and compute the gradients w.r.t. $\boldsymbol{z}$ or $\boldsymbol{\theta}$ to minimize the detection score. After updating $\boldsymbol{z}$ or $\boldsymbol{\theta}$, we re-render the object. We alternate between updating the parameters and re-rendering until we achieve a desired detection score.

Note that though the rendering pipeline involves beam casting with binary search, which is not a differentiable operation, once a specific point $\boldsymbol{x}_i$ is found and is affixed to the SHIFT3D object's surface, the rest of the pipeline is fully differentiable w.r.t. $\boldsymbol{z}$ and $\boldsymbol{\theta}$. We now discuss the differentiation process and loss function in detail.

3.3.1 Adversarial Loss Functions

To fool the detector, we define the adversarial loss:

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

where $\boldsymbol{y}^*$ is the inserted object's ground-truth bounding box, and IoU represents standard intersection over union function. A similar adversarial function has been used in [22] and [29]. Our adversarial loss function is designed to suppress all bounding box proposals that overlap with the ground truth, with the degree of suppression weighted by the amount of overlap. During optimization, we only back-propagate through the the detection confidence, p_i, to focus on minimizing the detection score rather than the bounding box parameters.

Given an input shape, $\boldsymbol{z}_0$, we regularize the perturbed shape, $\boldsymbol{z}$, to be close to $\boldsymbol{z}_0$. Specifically, we add an ℓ_2 regularization to $\mathcal{L}_{\text{adv}}$ and it prevents the perturbed shape from significantly diverging from the original shape.

$$\min_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}) = \min_{\boldsymbol{z}} \mathcal{L}_{\text{adv}}(\boldsymbol{z}) + \lambda \|\boldsymbol{z} - \boldsymbol{z}_0\|_2^2, \tag{2}$$

where λ is a hyper-parameter that controls the strength of regularization.

When optimizing object pose, $\boldsymbol{\theta}$, we apply hard constraints to restrict the $\boldsymbol{\theta}$ value within an ℓ_2 norm ball $\mathcal{B}(\boldsymbol{\theta}_0)$ centered at the input pose, $\boldsymbol{\theta}_0$. This hard constraint helps to ensure that the object does not move too far away from the baseline position. The optimization problem for the pose parameters can be written as,

$$\min_{\boldsymbol{\theta} \in \mathcal{B}(\boldsymbol{\theta}_0)} \mathcal{L}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathcal{B}(\boldsymbol{\theta}_0)} \mathcal{L}_{\text{adv}}(\boldsymbol{\theta}). \tag{3}$$

We note that during this process, any change made to the object's pose will also necessitate a corresponding modification of its ground truth y^* in $\mathcal{L}_{\text{adv}}$ at each step.

3.3.2 Optimization

To optimize the loss functions in Equations (2) and (3), we need to back-propagate through both the detector model, f, and the the object's SDF function, g. The gradients $d\mathcal{L}/d\boldsymbol{z}$ and $d\mathcal{L}/d\boldsymbol{\theta}$ can easily be computed using implicit differentiation, and are given by,

$$\frac{d\mathcal{L}}{d\boldsymbol{z}} = \sum_{i=1}^{n} m_i \frac{d\boldsymbol{x}_i}{d\boldsymbol{z}} \frac{d\mathcal{L}}{d\boldsymbol{x}_i}, \tag{4}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \sum_{i=1}^{n} m_i \frac{d\boldsymbol{x}_i}{d\boldsymbol{\theta}} \frac{d\mathcal{L}}{d\boldsymbol{x}_i}, \tag{5}$$

where the mask m_i has value of 1 if $\boldsymbol{x}_i$ is on object's surface and 0 otherwise. m_i helps remove any gradients from background points that do not contribute to the object's shape or pose. The gradients $d\boldsymbol{x}_i/d\boldsymbol{z}$ and $d\boldsymbol{x}_i/d\boldsymbol{\theta}$ are $d_{\boldsymbol{z}} \times 3$ and $d_{\boldsymbol{\theta}} \times 3$ Jacobian matrices, respectively. For the points on the object's surface, these gradients can easily be determined through implicit differentiation, since they satisfy the constraints $g(\boldsymbol{z}, \boldsymbol{x}_i) = 0$.

3.3.3 Gradients for Adversarial Shape

To compute $d\boldsymbol{x}_i/d\boldsymbol{z}$ in Eq. (4), we parameterize the LiDAR points as $\boldsymbol{x}_i = k_i \boldsymbol{e}_i + \boldsymbol{s}$, where $\boldsymbol{s} \in \mathbb{R}^3$ denotes the LiDAR sensor position, $\boldsymbol{e}_i \in \mathbb{R}^3$ is a unit vector in the direction of $\boldsymbol{x}_i$ positioned at $\boldsymbol{s}$, and k_i is the distance of the point from the sensor. With this parameterization, we can represent $\frac{d\boldsymbol{x}_i}{d\boldsymbol{z}} = \frac{dk_i}{d\boldsymbol{z}} \boldsymbol{e}_i^T$, and Eq. (4) can be written as,

$$\frac{d\mathcal{L}}{d\boldsymbol{z}} = \sum_{i=1}^{n} m_i \left(\boldsymbol{e}_i \cdot \frac{d\mathcal{L}}{d\boldsymbol{x}_i} \right) \frac{dk_i}{d\boldsymbol{z}}. \tag{6}$$

Since the points $\boldsymbol{x}_i$ on the object ($m_i = 1$) are subject to the constraint $g(\boldsymbol{z}, \boldsymbol{x}_i) = 0$, we can use implicit differentiation to calculate $dk_i/d\boldsymbol{z}$, and write Eq. (6) as,

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{z}} &= -\sum_{i=1}^{n} m_i \left(\boldsymbol{e}_i \cdot \frac{d\mathcal{L}}{d\boldsymbol{x}_i}\right)\left(\frac{\partial g}{\partial k_i}\right)^{-1} \frac{\partial g(\cdot, \boldsymbol{x}_i)}{\partial \boldsymbol{z}} \\ &= -\sum_{i=1}^{n} m_i \left(\boldsymbol{e}_i \cdot \frac{d\mathcal{L}}{d\boldsymbol{x}_i}\right)\left(\boldsymbol{e}_i \cdot \frac{\partial g(\boldsymbol{z}, \cdot)}{\partial \boldsymbol{x}_i}\right)^{-1} \frac{\partial g(\cdot, \boldsymbol{x}_i)}{\partial \boldsymbol{z}}.\end{aligned} \tag{7}$$

3.3.4 Gradients for Adversarial Pose

Similar to shape gradient, $dk_i/d\boldsymbol{z}$, we can compute $dk_i/d\boldsymbol{\theta}$ and use implicit differentiation to update Eq. (5) as,

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \sum_{i=1}^{n} m_i \left(\frac{d\boldsymbol{s}}{d\boldsymbol{\theta}} + k_i \frac{d\boldsymbol{e}_i}{d\boldsymbol{\theta}} + \frac{dk_i}{d\boldsymbol{\theta}} \boldsymbol{e}_i^T\right) \frac{d\mathcal{L}}{d\boldsymbol{x}_i}. \\ &= \sum_{i=1}^{n} m_i \left[\frac{d\boldsymbol{s}}{d\boldsymbol{\theta}} + k_i \frac{d\boldsymbol{e}_i}{d\boldsymbol{\theta}} - \left(\frac{\partial g}{\partial k_i}\right)^{-1} \frac{\partial g}{\partial \boldsymbol{\theta}} \boldsymbol{e}_i^T\right] \frac{d\mathcal{L}}{d\boldsymbol{x}_i} \\ &= \sum_{i=1}^{n} m_i \left(\frac{d\boldsymbol{s}}{d\boldsymbol{\theta}} + k_i \frac{d\boldsymbol{e}_i}{d\boldsymbol{\theta}}\right) \left[\mathbf{I} - \left(\boldsymbol{e}_i \cdot \frac{\partial g}{\partial \boldsymbol{x}_i}\right)^{-1} \frac{\partial g}{\partial \boldsymbol{x}_i} \boldsymbol{e}_i^T\right] \frac{d\mathcal{L}}{d\boldsymbol{x}_i}.\end{aligned} \tag{8}$$

We present the overall procedure for adversarial shape and pose generation in Algorithm 1. Note that for pose generation, after gradient descent, we also need to project the resulting $\boldsymbol{\theta}$ back to $\mathcal{B}(\boldsymbol{\theta}_0)$ to satisfy the hard constraints. Details of calculating gradients in (7) and (8) are discussed in Section A.3 of the supplementary materials.

Algorithm 1 Pseudocode for adversarial perturbations of an object's shape or pose in a given LiDAR scene.

Input: Object SDF model $g(\cdot, \cdot)$, initial shape, $\boldsymbol{z}_0$, or pose parameters, $\boldsymbol{\theta}_0$, detector model f, LiDAR sensor positions $\boldsymbol{s}$, input scene $\{\hat{\boldsymbol{x}}_i\}$, maximum iterations N_{iter}, and learning rate α.

Output: Adversarial shape, $\boldsymbol{z}_{\text{adv}}$, or pose, $\boldsymbol{\theta}_{\text{adv}}$, parameters.

Calculate LiDAR beam length $\hat{k}_i \leftarrow \|\hat{\boldsymbol{x}}_i - \boldsymbol{s}\|_2$.
Calculate LiDAR beam directions $\boldsymbol{e}_i \leftarrow (\hat{\boldsymbol{x}}_i - \boldsymbol{s})/\hat{k}_i$.
$\boldsymbol{z} \leftarrow \boldsymbol{z}_0$ or $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$; $\mathcal{L}_{\text{min}} \leftarrow \infty$.
Render object: calculate rendered points $\{\boldsymbol{x}_i\}$, and set $m_i \leftarrow 1$ when points on the object. (Section 3.2).
while iteration $< N_{\text{iter}}$ **do**
    Calculate $d\mathcal{L}/d\boldsymbol{z}$ using Eq. (7) or $d\mathcal{L}/d\boldsymbol{\theta}$ using Eq. (8).
    $\boldsymbol{z} \leftarrow \boldsymbol{z} - \alpha \cdot d\mathcal{L}/d\boldsymbol{z}$ or $\boldsymbol{\theta} \leftarrow \text{Proj}_{\mathcal{B}(\boldsymbol{\theta}_0)}[\boldsymbol{\theta} - \alpha \cdot d\mathcal{L}/d\boldsymbol{\theta}]$.
    Re-render object with the updated $\boldsymbol{z}$ or $\boldsymbol{\theta}$ in the input scene and update $\{\boldsymbol{x}_i\}$ and m_i.
    Forward propagate to obtain $\mathcal{L}$.
    if $\boldsymbol{x}_i$ satisfy all realism constraints **then**
        $\mathcal{L}_{\text{min}} \leftarrow \min\{\mathcal{L}_{\text{min}}, \mathcal{L}\}$.
        if $\mathcal{L} = \mathcal{L}_{\text{min}}$ **then**
            $\boldsymbol{z}_{\text{adv}} \leftarrow \boldsymbol{z}$ or $\boldsymbol{\theta}_{\text{adv}} \leftarrow \boldsymbol{\theta}$.

4. Experiments

We first describe our datasets and models, followed by an overview of our experimental setup. We then dive into results, highlighting that SHIFT3D outputs are effective at confusing detectors and robust along various axes, while demonstrating the potential insights SHIFT3D can provide for improving detection models in the real world.

4.1. Datasets and Models

We utilize the Waymo Open Dataset (WOD) [20], which provides LiDAR point clouds and 3D bounding box annotations of objects detected by LiDAR sensors mounted on autonomous vehicles. The large scale and high quality of WOD, coupled with its extensive geographical coverage, make it an excellent benchmark for evaluating a variety of scenarios. Following the approach of [22], we focus on the "Vehicle" category in WOD and restrict our evaluation to 2D bounding boxes in a bird's eye view.

Our object detection models are based on the PointPillars architecture [13] and SST [8], which operate solely on point cloud data without utilizing auxiliary inputs such as intensity. Specifically, PointPillars partitions the input points into discrete bins (i.e., pillars) from a bird's eye view, and for each pillar, extracts features using PointNet. SST uses a single-stride sparse transformer to maintain the original resolution from the beginning to the end of the network. We trained our model on the WOD training set and use the validation set of WOD as input scenes for rendering objects. On the vanilla detection task, our PointPillars and SST models achieve comparable performance with the baseline in [20] (refer to Section B of the supplementary material for more on baseline detection metrics, such as AP and APH).

Our DeepSDF model is trained on the "Automobile" category of ShapeNet [4]. We used the same training procedure and model architecture as the original DeepSDF paper [18], where the latent shape encoding $\boldsymbol{z}$ has a dimension of $d_{\boldsymbol{z}} = 256$. We selected 5 representative vehicle objects from the "Automobile" category as our baseline objects: Coupe, Sports Car, SUV, Convertible, and Beach Wagon. The meshes of these objects reconstructed by DeepSDF are depicted in the third column of Figure 7. To ensure that the dimensions of each object are realistic and consistent with the natural vehicles in the scene, we manually scaled each baseline object.

4.2. Setup

We insert our baseline SDF objects into 500 randomly chosen WOD validation set scenes. To ensure diversity in object placement, we randomly positioned the inserted objects between 15 and 50m from the autonomous vehicle.

The object's heading was randomly chosen between 0 and 2π. Keeping objects at least 15m away reduces the number of points rendered on the object and avoids out-of-memory issues. To describe the pose transformations, we use a 6-dimensional vector $\boldsymbol{\theta}$ consisting of the coordinates (x, y, z) and the angles of yaw, pitch, and roll.

We draw a sphere with a 7m radius centered at the location of the inserted object. If we emit a beam from the LiDAR source to any point, and that beam lies completely outside this sphere, we ignore that point as it is far enough away from the inserted object to not affect the rendering process. For additional implementation details, please refer to Section A in supplementary material.

4.3. Hyper-parameters

We run 40 steps of adversarial perturbation. During optimization, we record the result with lowest loss that also satisfies all constraints specified in Section 3.2.1. We pick a learning rate of 0.01 for all experiments after extensive hyperparameter search. In adversarial shape optimization, we select a λ of either 1 or 10, depending on which results in the lowest loss value $\mathcal{L}_{\text{adv}}(\boldsymbol{z})$. However, in cases where different settings of λ produce very similar loss values, we will choose the larger λ as it keeps the perturbed shape closer to the original. Refer to Section B of the supplementary material for detailed discussion on hyper-parameters. To achieve realistic object placement, we set $\epsilon_{\text{overlap}} = -0.02$ and $\epsilon_{\text{float}} = 0.02$, which are defined in Section 3.2.1. Additionally, we exclude cases where the inserted SDF objects may be occluded by other objects in the scene, by disallowing inserted objects that result in fewer than 300 on-object points post rendering. Each object is placed randomly with the above realism constraints into 500 scenes. Note that our focus is on the detector's performance under varying conditions, including rare ones. So we consciously opted not to enforce additional traffic constraints we placing the inserted objects.

	Area Under Curve (AUC)				
Method	Coupe	Sports Car	SUV	Conv. Car	Beach Wagon
Baseline	0.755	0.756	0.755	0.681	0.779
Random	0.716	0.721	0.728	0.625	0.746
SHIFT3D +noise	0.469	0.451	0.473	0.392	0.519
SHIFT3D	**0.466**	**0.448**	**0.469**	**0.390**	**0.514**

Table 1: The AUC for the curves in Figure 3, which demonstrates that SHIFT3D produces challenging examples that confuse a 3D detector far more successfully than random perturbations. Small perturbations of SHIFT3D objects do not appreciably affect their ability to confuse detectors.

4.4. Evaluation Metrics

We consider an inserted 3D object to be detected if a predicted bounding box with detection score greater than a threshold exists with box center less than 5m from the center of the added object. If multiple boxes meet these criteria, we select the box with the highest detection score. To assess the performance of the detector, we generate a *threshold-recall curve* by sweeping the detection score threshold from 0 to 1 and calculating the recall rate for the added SDF objects (the precision of detecting inserted objects is always 100%), with AUC being then calculated for each curve as a useful overall detector performance metric. Refer to the supplementary material for detailed discussion on metrics.

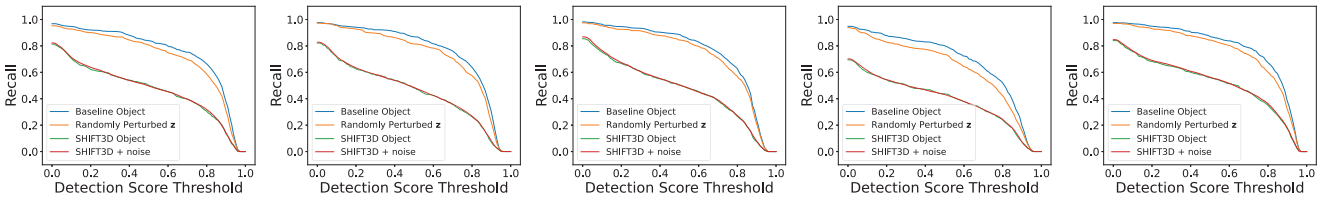
4.5. Adversarial Shape Generation Results

We first present our experimental results on PointPillars, while the results on SST are shown in Section E of the supplementary material.

Figure 3 presents the threshold-recall curve for each baseline object and its corresponding adversarial objects across the 500 scenes. Since the generated adversarial objects are scene-specific, their shapes vary across different scenes. To quantify the overall reduction in recall, we compute the area under the threshold-recall curves (AUC) and report the numerical values in Table 1.

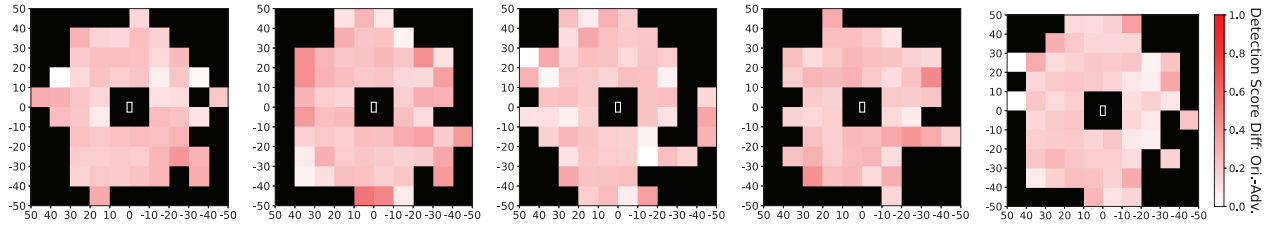
We compare the performance of our method against objects generated by randomly perturbing the latent shape encoding. For each adversarial object, we compute the ℓ_2 distance between the original encoding $\boldsymbol{z}_0$ and the encoding $\boldsymbol{z}_{\text{adv}}$ generated by SHIFT3D, and generate another object by randomly selecting another encoding $\boldsymbol{z}_{\text{rand}}$ on a $d_{\boldsymbol{z}}$-dimensional sphere with a radius of $\|\boldsymbol{z}_0 - \boldsymbol{z}_{\text{adv}}\|_2$. The results presented in Figure 3 and Table 1 clearly show that the recall rate of the adversarial objects is significantly lower compared to both the randomly perturbed and baseline objects. These findings demonstrate that our method is substantially more effective at generating adversarial samples. To analyze the robustness of SHIFT3D outputs, we sample 10 additional $\boldsymbol{z}$ encodings around $\boldsymbol{z}_{\text{adv}}$ by adding Gaussian noise with standard deviation 0.01, which is approximately 5% of the average value of $\|\boldsymbol{z}_0 - \boldsymbol{z}_{\text{adv}}\|_2$. As shown in Figure 3 and Table 1, such examples have almost identical Threshold-recall curves and area under curve values as those of $\boldsymbol{z}_{\text{adv}}$.

To assess the transferability of SHIFT3D outputs across different locations in a scene, we randomly placed each of the 500 objects in 10 distinct scenes while satisfying the physical placement constraints described in Section 3.2.1. We applied random yaw rotations during the placements and maintained consistency by subjecting the baseline objects to identical rotation and translation operations. We then divided the resulting adversarial and baseline objects into $10m \times 10m$ cells and computed the average reduction



(a) Coupe (b) Sports Car (c) SUV (d) Convertible Car (e) Beach Wagon

Figure 3: Threshold-recall curves to evaluate adversarial *shape* generation for different vehicles in the "Automobile" category. Each curve represents the recall rate of the detector at different detection threshold values, computed from 500 scenes with the corresponding vehicle present. The proposed method for generating adversarial shapes shows significantly lower recall rates compared to random perturbation, demonstrating its effectiveness in deceiving the detector.



(a) Coupe (b) Sports Car (c) SUV (d) Convertible Car (e) Beach Wagon

Figure 4: Evaluation of the robustness of adversarial shapes generated for all vehicles across various poses and locations in the scene from a bird's eye view. The plot depicts the detection score reduction compared to the original shape in the same pose, where black cells indicate no placement. The rectangle at the center represents the autonomous vehicle. This figure illustrates the transferability and robustness of the adversarial shapes across different locations in the input scene scene.

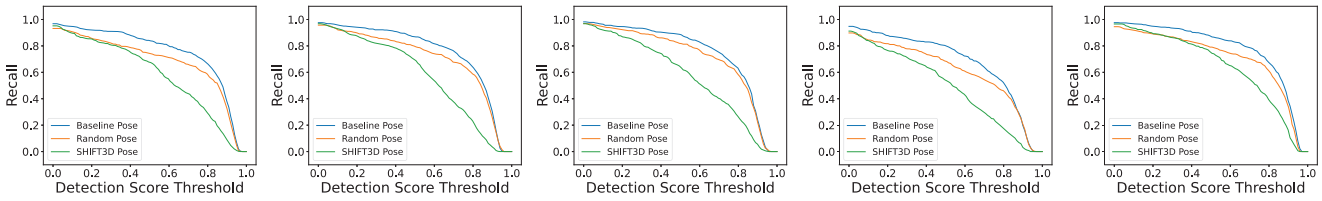
in detection scores. As shown in Figure 4, we found that the detection scores of the adversarial objects consistently decreased across all locations, indicating that the adversarial shapes are not location-specific and remain robust when placed in different positions in a scene.

In Figure 7 we present visualizations for the challenging objects generated by SHIFT3D, together with their corresponding point cloud scenes. We also point out the semantic patterns of the SHIFT3D objects. Crucially, as we visualize more SHIFT3D objects, we can begin to detect semantic patterns; for example, vehicles presenting with lower chassis will very often fool the 3D detector. These insights highlight a key benefit of using SHIFT3D, as we now have concrete candidates for proactive model improvements. To validate these semantic features present in the log data, we identify and test natural objects in the WOD log data that are most similar to the objects generated by SHIFT3D. Those objects are retrieved by reconstruct object shapes with DeepSDF and select the object whose latent shape vector z_{natural^*} is the closest one to a SHIFT3D generated latent shape vector z_{SHIFT3D}. Detailed information about this experiment is presented in Section F of the supplementary material.

Visualization of more objects generated by SHIFT3D is presented in Section C of the supplementary material. In Figure 10 of the supplementary, we also present visualizations of adversarial objects and the corresponding detection scores at various steps of the optimization process in SHIFT3D. We also test SHIFT3D on an SST detector and our results are presented in Section E in the supplementary materials. Additionally, we report transferability of SHIFT3D between SST and PointPillars models in Section E.3.

4.6. Adversarial Pose Generation Results

We present threshold-recall curves for both original and SHIFT3D poses of objects across 500 scenes in Figure 5, along with comparisons to randomly generated poses. For pose generations in both SHIFT3D and random, we limit displacement to under 4 meters on the xy-plane. We manually verified that a buffer zone of 3 meters around the object is enough to prevent any part of it from exceeding the the 7m sphere we draw around the object as in Section 4.2. We also limit rotations to under ± 0.1 rads on the pitch and roll axes, and do not limit rotation along the yaw axis, while ensuring that the constraints in Section 3.2.1 are satisfied. Our results, presented in Figure 5 and Table 2, demonstrate a significant reduction in recall performance for adversar-



(a) Coupe (b) Sports Car (c) SUV (d) Convertible Car (e) Beach Wagon

Figure 5: Threshold-recall curves evaluate adversarial *pose* for "Automobile" category vehicles. Similar to the curves in Figure 3, each curve shows the detector recall rate at different thresholds, from 500 scenes with the vehicle present. Our method exhibits significantly lower recall rates, outperforming random pose in deceiving the detector.

Method	Area Under Curve (AUC)				
	Coupe	Sports Car	SUV	Conv. Car	Beach Wagon
Baseline	0.755	0.756	0.755	0.681	0.779
Random	0.683	0.705	0.719	0.611	0.713
SHIFT3D	**0.580**	**0.574**	**0.571**	**0.489**	**0.648**

Table 2: AUC metric of the curves in Figure 5 quantifies the proposed method's improvements in adversarial *pose* generation. Random perturbations have little effect on detection score, while our method results in lower AUC values for all vehicles, successfully deceiving detection performance.

ial objects, even when placed in alternate poses. In contrast, random pose perturbations with the same constraints result in a minimal reduction in recall. Visualizations for SHIFT3D adversarial pose generation are presented in Section D in the supplementary material.

4.7. Improving Model Robustness Using Objects Generated by SHIFT3D

We also conducted a pilot study which aims to investigate the effectiveness of using objects generated by SHIFT3D for data augmentation and improving the robustness of our detector model. We randomly select 2000 scenes containing objects generated by our method from our five baseline vehicles and use them to fine-tune our PointPillar model with a learning rate of 10^{-7} for one epoch. We evaluate the performance of the detector model by testing it on another 500 scenes, each containing an object from the 5 baseline vehicles. We compare the performance of the finetuned detector model under SHIFT3D against the vanilla detector and plot the threshold-recall curve in Figure 6. Additionally, we report the AUC values in Table 3. Our results show a clear improvement in the AUC metric for both baseline and SHIFT3D generated objects. Additionally, we evaluate the performance of the fine-tuned model on the vanilla WOD detection task and present the results in

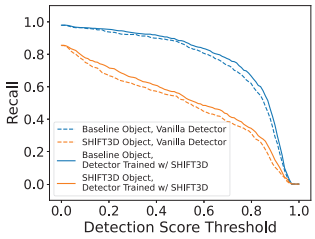


Figure 6: Threshold-recall curves before and after fine-tuning the 3D detector with objects generated by SHIFT3D.

Table 4 in Section B.1 of the supplementary material. Our result reveals that the performance metrics of the fine-tuned model are nearly identical to those of the vanilla detector when evaluated on the natural WOD data. This suggests that our fine-tuning approach does not introduce any significant regression. Furthermore, we may observe a slight improvement in the model's performance on the natural data, although this effect, if present, appears to be rather small.

Models	Area Under Curve (AUC)	
	Vanilla Detector	Finetuned Detector
Baseline	0.751	**0.774**
SHIFT3D	0.482	**0.514**

Table 3: AUC metric values before and after fine-tuning the 3D detector with objects generated by SHIFT3D.

5. Conclusion

SHIFT3D is a differentiable pipeline for generating challenging yet natural examples that provide preemptive insights into failure modes in 3D vision systems. We demonstrated that all model inputs generated by SHIFT3D reliably confuse 3D detectors in an autonomous vehicles setting, and do so regardless of the insertion location or reference scene. Through diagnostic information provided by SHIFT3D, we will be better equipped to catch failures within 3D vision models before they appear, and circumvent scenarios where single missed objects lead to catastrophic results.

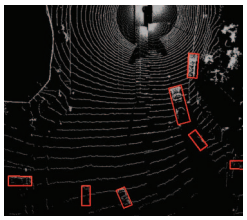
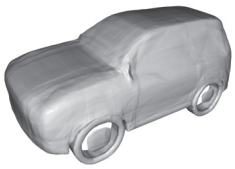
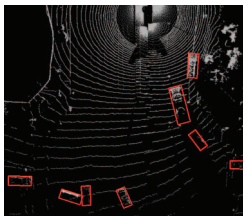
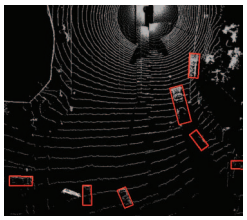
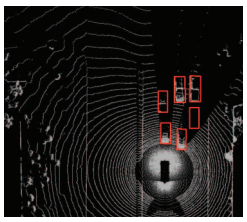
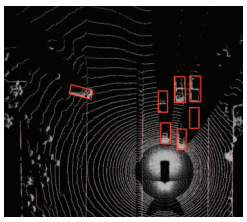
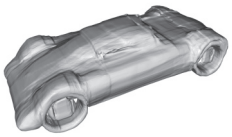
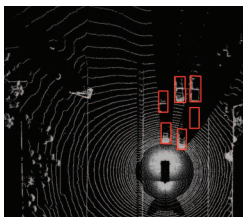
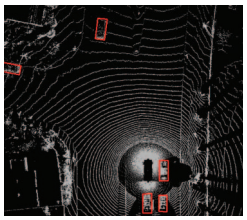
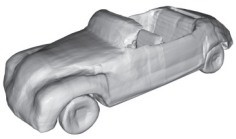
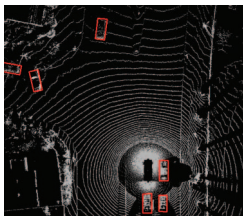
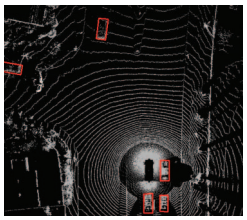
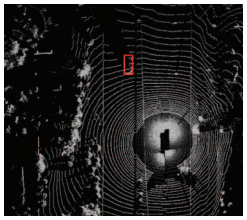
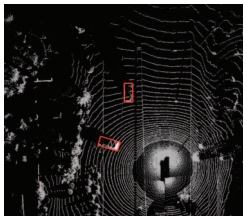
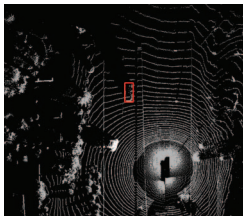
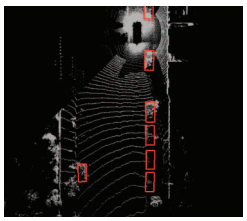
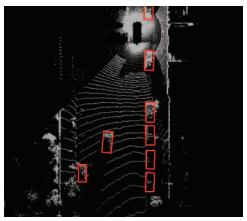
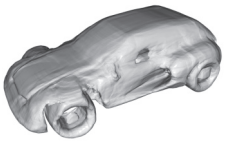
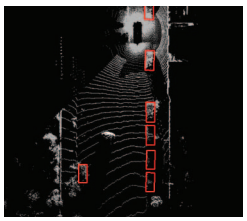
Input Scene	Baseline Object	Baseline Object in the Scene	SHIFT3D Object and Their Semantic Patterns	SHIFT3D Object in the Scene
	 SUV	 Detection Score: 0.74	 smaller windshield	 Detection Score: 0.07
	 Sports Car	 Detection Score: 0.91	 lower chassis square front	 Detection Score: 0.04
	 Convertible Car	 Detection Score: 0.71	 lower chassis round front	 Detection Score: 0.07
	 Beach Wagon	 Detection Score: 0.93	 shorter chassis hatchback	 Detection Score: 0.10
	 Coupe	 Detection Score: 0.93	 shorter chassis wider	 Detection Score: 0.22

Figure 7: Visualizations of challenging objects created by SHIFT3D and their scenes. Red boxes are detector's output.

References

[1] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4845–4854, 2019. 2

[2] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021. 2

[3] 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. 2

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[5] Zhao Chen, Vincent Casser, Henrik Kretzschmar, and Dragomir Anguelov. Gradtail: Learning long-tailed data using gradient-based sample weighting. *arXiv preprint arXiv:2201.05938*, 2022. 1

[6] Zeyad Ali Sami Emam, Hong-Min Chu, Ping-Yeh Chiang, Wojciech Czaja, Richard Leapman, Micah Goldblum, and Tom Goldstein. Active learning at the imagenet scale. *arXiv preprint arXiv:2111.12880*, 2021. 1

[7] Francis Engelmann, Jörg Stückler, and Bastian Leibe. Samp: shape and motion priors for 4d vehicle reconstruction. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 400–408. IEEE, 2017. 17

[8] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8458–8468, 2022. 5

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[10] Chiyu Max Jiang, Mahyar Najibi, Charles R Qi, Yin Zhou, and Dragomir Anguelov. Improving the intra-class long-tail in 3d detection via rare example mining. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, pages 158–175. Springer, 2022. 1

[11] Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4773–4783, 2019. 2

[12] Jaeyeon Kim, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Minimal adversarial examples for deep learning on 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7797–7806, 2021. 2

[13] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 5

[14] Daniel Liu, Ronald Yu, and Hao Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283. IEEE, 2019. 2

[15] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3d point clouds. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 88–104. Springer, 2020. 2

[16] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2537–2546, 2019. 1

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2

[18] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2, 3, 5

[19] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 5, 12, 13, 15

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[22] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020. 2, 4, 5, 17

[23] Rahul Venkatesh, Eric Wong, and J Zico Kolter. Semantic adversarial robustness with differentiable ray-tracing. *Workshop on Differentiable Vision, Graphics, and Physics in Machine Learning at NeurIPS 2020*, 2020. 2

[24] Matthew Wicker and Marta Kwiatkowska. Robustness of 3d deep learning in an adversarial setting. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11767–11775, 2019. 2

[25] Tsung-Han Wu, Yueh-Cheng Liu, Yu-Kai Huang, Hsin-Ying Lee, Hung-Ting Su, Ping-Chia Huang, and Winston H. Hsu. Redal: Region-based and diversity-aware active learning for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15510–15519, October 2021. 1

[26] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019. 2

[27] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018. 2

[28] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6898–6907, 2019. 2

[29] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1369–1378, 2017. 2, 4

[30] 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

[31] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4302–4311, 2019. 2

[32] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1598–1606, 2019. 2

[33] Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and Nenghai Yu. Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10356–10365, 2020. 2