Supplementary Material – 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

In this supplementary material we first provide details on how to render a triangle mesh into a point cloud in a differentiable manner. Next, we present a brief overview of the qualitative video results included in supplementary materials. Several frames are also included in this document.

1. Ray Casting

In this section, we discuss details how to compute the intersection of rays and triangles analytically using the Moller Trumbore algorithm. Consider a triangle with vertices $ABC \in \mathbb{R}^3$ and a ray with origin $O \in \mathbb{R}^3$ and direction $D \in \mathbb{R}^3$ shown in Figure 1. We can write the intersection of the ray and the plane of $\triangle ABC$ as:

$$P = wA + uB + vC = O + tD \tag{1}$$

where $u, v, w, t \in \mathbb{R}$ and

$$u + v + w = 1 \tag{2}$$

forces P to lie on the plane of $\triangle ABC$.

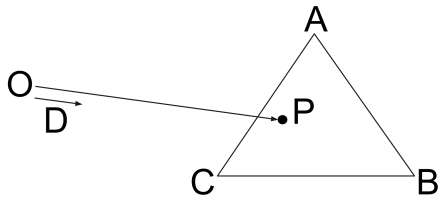

Figure 1: Computing the intersection of a ray and a triangle. We label triangle vertices as A, B, C, parameterize the ray with origin O and directional unit vector D, and denote the intersection point as P

Here u, v, w represent barycentric coordinates with respect to $\triangle ABC$ and t determines the distance between O and P. In our work, we set $||D||_2 = 1$ so that t measures euclidean distance. To compute the intersection, we define $w = 1 - u - v$ and solve for t, u, v. In addition, define

$$E_1 = B - A$$
$$E_2 = C - A$$

to denote edges AB and AC for simplicity. Then t, u, v can be obtained as follows

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{D \cdot (E_1 \times E_2)} \begin{bmatrix} (A - O) \cdot (E_1 \times E_2) \\ (D \times E_2) \cdot (A - O) \\ (A - O) \cdot (E_1 \times D) \end{bmatrix}. \tag{3}$$

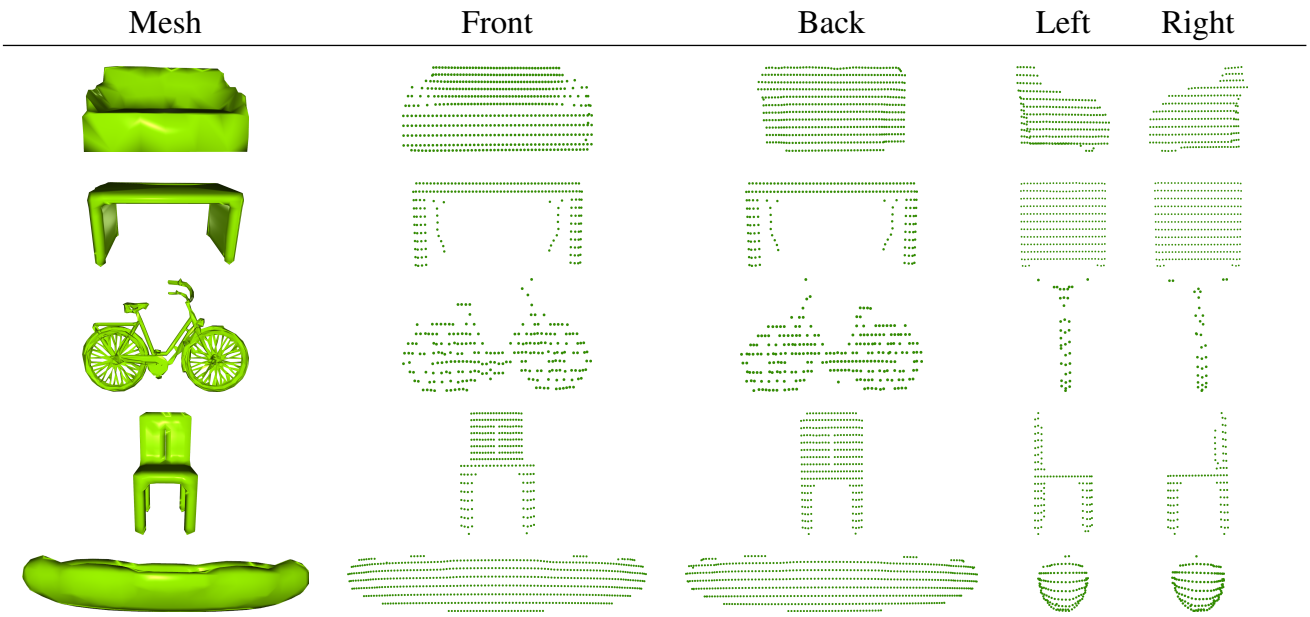

Table 1: Examples of rendering various different objects from different viewpoints. In this table the objects are placed 5 meters away from the camera.

We then check the condition

$$P \; in \; \triangle ABC \iff u, v, w \in [0, 1] \tag{4}$$

to see if P lies inside $\triangle ABC$ to determine if the ray intersects with the triangle.

This method calculates the intersections precisely and is differentiable. When considering multiple rays and faces during rendering, we calculate the intersection between all pairs of rays and triangles and take the closest intersection for each ray to be a hit. These operations can be efficiently implemented with tensor operations to run on a GPU. Some examples of rendering objects from various angles are shown in Table 1.

2. Additional Qualitative Results

Outside of this document, we present visualizations of our attack on entire snippets in the video *supplementary_video.mp4*. Here, we provide a short summary of these results and provide several frames from the video snippets in Table 2 for those who cannot view our video.

2.1. Attack Demonstration

To demonstrate the universal nature of the adversary, we visualize the attack over hundreds of frames. In these videos we compare detection on original snippets from KITTI against detection on perturbed snippets. In the perturbed snippets, the adversarial mesh is placed on a host vehicle and hides the host vehicle with high probability from various poses, showing its robustness to motion. Furthermore, we present such results on several snippets with different host vehicles and traffic scenarios to illustrate that the adversarial object works in a variety of settings.

2.2. Defense Demonstration

We also include in *supplementary_video.mp4* qualitative results for our proposed defense to show that data augmentation successfully defends our attack. For fair evaluation of defense, we retrain the adversarial mesh to attack the defended model. In these videos, we visualize perturbed snippets and compare the original detection model with a detection model trained with data augmentation.

2.3. Mesh Deformation

Finally, some visualizations of mesh deformation during learning are shown in Table 3.

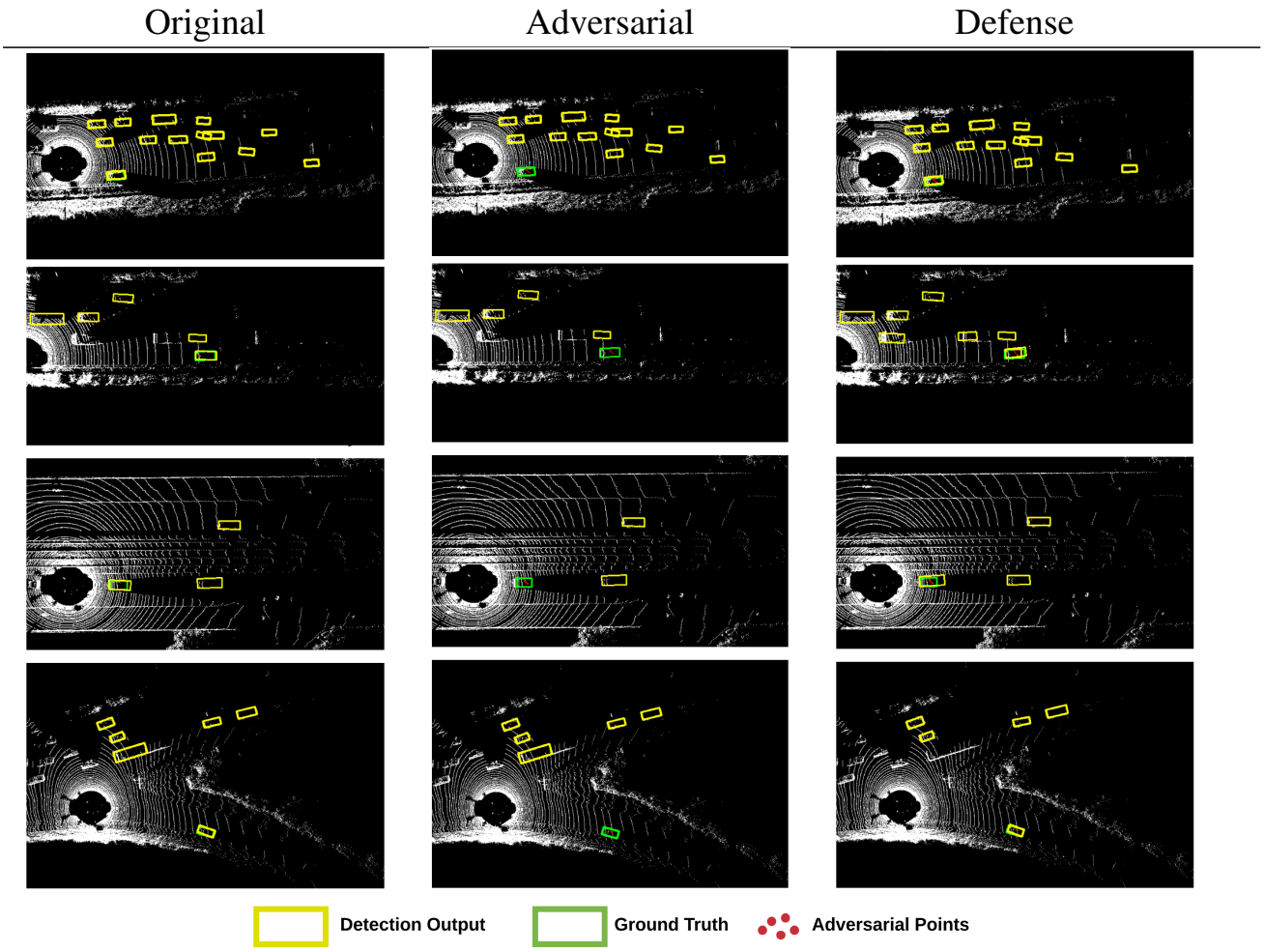

Table 2: Qualitative demonstrations of our attack and defense. The left column displays detection on frames from KITTI, the middle column displays detection on perturbed frames, and the right column shows results of attacking a defended model.

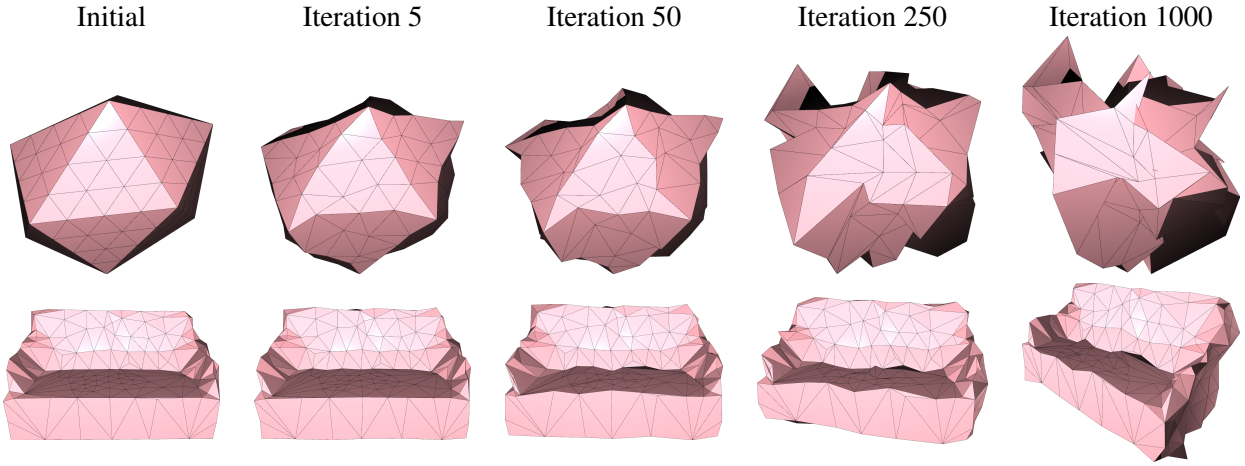

Table 3: Mesh deformation during optimization visualized at iterations 0, 5, 50, 250, 1000. On the first row we show the deformation of an icosphere into an adversarial mesh. In the second row, we learn an adversary that looks like a couch. Here the l_∞ norm of vertex perturbations is constrained but we allow free rotation.