REALISTIC ADVERSARIAL EXAMPLES IN 3D MESHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Highly expressive models especially deep neural networks (DNNs) have been widely applied to various applications and achieved increasing success. However, recent studies show that such machine learning models appear to be vulnerable against adversarial examples. So far adversarial examples have been heavily explored for 2D images, while few work has tried to understand the vulnerabilities of 3D objects which exist in real world, where 3D objects are projected to 2D domains by photo taking for different learning (recognition) tasks. In this paper we consider adversarial behaviors in practical scenarios by manipulating the shape and texture of a given 3D mesh representation of an object. Our goal is to project the optimized "adversarial meshes" to 2D with photo-realistic rendering engine, and still able to mislead different machine learning models. Extensive experiments show that by generating unnoticeable 3D adversarial perturbation on shape or texture for a 3D mesh, the corresponding projected 2D instance can either lead classifiers to misclassify the victim object arbitrary malicious target, or hide any target object within the scene from state-of-the-art object detectors. We conduct human studies to show that our optimized adversarial 3D perturbation is highly unnoticeable for human vision systems. In addition to the subtle perturbation on a given 3D mesh, we also propose to synthesize a realistic 3D mesh to put in a scene mimicking similar rendering conditions and therefore attack existing objects within it. In-depth analysis for transferability among different 3D rendering engines and vulnerable regions of meshes are provided to help better understand adversarial behaviors in practice and motivate potential defenses.

1 INTRODUCTION

Machine learning, especially deep neural networks, have achieved great successes in various domains, including image recognition (He et al., 2016), natural language process (Collobert & Weston, 2008), speech to text translation (Deng et al., 2013), and robotics (Silver et al., 2016). Despite the increasing successes, machine learning models are found vulnerable to adversarial examples. Small magnitude of perturbation is added to the input, such as an image, and therefore different learning models can be misled to make targeted incorrect prediction. Such adversarial examples have been widely studied in 2D domain (Szegedy et al., 2013; Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016; Carlini & Wagner, 2017; Xiao et al., 2018a;b) , while in practice directly manipulating pixel values of real world observations is hard. As a result, it is important to explore the vulnerabilities of 3D meshes in practice. In addition, synthetic datasets have been widely used to improve the performance of machine learning models for various tasks, including view point estimation (Su et al., 2015), semantic understanding in street views (Richter et al., 2016), human pose estimation (Varol et al., 2017; Chen et al., 2015), 3D shape reconstruction (Yang & Deng, 2018; Massa et al., 2016), and indoor scene understanding (Song et al., 2017; Zhang et al., 2017; Handa et al., 2016; McCormac et al., 2017). In these tasks, images and videos are captured through a physically-based rendering system, and such synthetic data are usually generated with different scene configurations and viewpoints based on the 3D assets are from large-scale datasets such as ShapeNet (Chang et al., 2015), ModelNet (Z. Wu, 2015) and SUNCG (Song et al., 2017). Therefore, it is critical to explore the possibilities of manipulating such 3D shape dataset and the corresponding severe consequences when rendering the "adversarial 3D meshes" for learning tasks.

Physical adversarial examples are studied (Kurakin et al., 2016; Evtimov et al., 2017; Eykholt et al., 2018; Athalye & Sutskever, 2017) but they do not focus on the 3D object itself or the real-world rendering process. In this paper, we propose to generate adversarial perturbation for 3D meshes by manipulating the shape or texture information, which can be eventually rendered to 2D domain to attack different machine learning models. We also propose to place a 3D mesh (here we use a bunny) rendered in physically based scenes to fulfill adversarial goals. We target to attack both classifiers (Inception-v3 by Szegedy et al. and DenseNet by Huang et al.) and object detectors (Yolo-v3 by Redmon & Farhadi). Our proposed 3D mesh based attack pipeline is shown in Figure 1. In particular, we leverage a physically-based rendering engine, which computes screen pixel values by raycasting view directions and simulating the shape and light interactions with physics models, to project 3D scenes to 2D domains. First, we propose to either generate perturbation for the shape or texture of a 3D mesh, and guarantee the rendered 2D instance can be misclassified by traditional classifiers into the targeted class. Here we do not control the rendering conditions (e.g. lighting and viewpoints) and show that the "3D adversarial mesh" is able to attack the classifier under various rendering conditions with almost 100% attack success rate. Second, we generate adversarial perturbation for 3D meshes to attack object detector in a synthetic indoor scene and show that by adding a bunny with subtle perturbation, the detector can mis-detect various existing objects within the scene. Third, we also propose to place a 3D mesh in a random outdoor scene and render it under similar physical conditions with existing objects to guarantee its realistic observation. We show that such added object can lead object detectors to miss the target object within the given real-world scene. To better evaluate adversarial perturbation on 3D meshes, we propose to use a smoothing loss (Vogel & Oman, 1996) as measurement metric for shape and report the magnitude of adversarial perturbation in various settings (best, average, worst) to serve as a baseline for future possible adversarial attacks on 3D meshes. We conduct user study to allow real humans to identify the categories of the generated adversarial 3D meshes, and the collected statistical results show that users recognize the adversarial meshes as ground truth with probability $99.29 \pm 1.96\%$.

In addition, we analyze transferability upon different types of renders, where perturbation against one render can be transferred to the other. We show that transferability among different renderers is high for untargeted attack but low for targeted attacks, which provides in-depth understanding of properties of different rendering system. The transferability makes black-box attacks possible against different rendering systems. For instance, attacker can attack the differentiable render and transfer the perturbation to non-differentiable ones which encounter high computational cost. In our experiments, we show that we can attack a differentiable render, the neural mesh renderer (Kato et al., 2018) and transfer the perturbation to the non-differetiable renderer Mitsuba (Jakob, 2010). Finally, to better understand the attacks the corresponding vulnerable regions for 3D meshes, we also analyze the manipulation flow for shape based perturbation, and find that the vulnerable regions of 3D meshes usually lie on the parts that are close to the viewpoint with large curvatures. This leads to better understanding of adversarial behaviors for real-world 3D objects and provide potential directions to enhance the model robustness, such as design adaptive attention mechanism or leverage deformation information to improve machine learning models.

In summary, our *contributions* are listed below: 1). We propose to generate adversarial perturbation on shape or texture for 3D meshes and use a physically-based rendering system to project such adversarial meshes to 2D and therefore attack existing classifiers under various rendering conditions with 100% attack success rate; 2). We propose to place a 3D mesh and put it in both indoor and outdoor scenes mimicking the same physical rendering conditions, and show that existing objects can be missed by object detectors; 3). We evaluate adversarial 3D meshes based on a 3D smoothing loss and provide a baseline for adversarial attacks on 3D meshes; 4). We evaluate the transferability for adversarial 3D meshes among different rendering systems and show that untargeted attack can achieve high transferability; 5). We propose a pipeline for blackbox attack that aims to attack a differentiable renderer and transfer the perturbation to a non-differentiable renderer in an unknown environment, by adding perturbation to the estimated environment to improve robustness. 6). We provide in-depth analysis for the vulnerable regions for 3D meshes and therefore lead to discussion for potential defenses.

2 3D ADVERSARIAL MESHES

2.1 PROBLEM DEFINITION

Let g be a machine learning model trained in 2D domain. I denotes a 2D image and its corresponding label is y. g aims to learn a mapping from image domain X to Y where $I \in X$ and $y \in Y$. A physically based rendering R projects a 2D image $I = R(S; P, L)$ from a 3D mesh S with camera parameters P and illumination parameters L. A 3D mesh S can be further represented as triangular meshes which consists of vertices V, texture T and faces F. An attacker aims to generate a "3D adversarial mesh" S^{adv} by manipulating the shape (vertices) and texture information for the original 3D mesh$S$, which is eventually rendered to a 2D image to mislead a machine learning model g so that $g(R(S^{\mathrm{adv}}; P, L)) \neq y$ (untarget attack) or $g(R(S^{\mathrm{adv}}; P, L)) = y'$ (target attack) where y' is the malicious target output and y is the ground truth.

Achieving the above goals is non-trivial with the following challenges. 1. 3D space to 2D space is complicated: a) 2D image space is largely reduced because we parameterize 2D images as the production from 3D shapes/textures/illumination, and such reduction can affect adversarial behaviors; b) 3D properties (shape/texture/illumination) are entangled together to generate the pixel values in a 2D image, so perturbing one will affect the other. c) This process in general is not differentiable unless we make substantial assumptions for the pipeline. 2. 3D space itself is complicated: a) 3D constraints such as physically possible shape geometry and texture are not directly reflected on 2D (Zeng et al., 2017). b) Human perception on objects are based on 3D understandings. Changes of 2D pixel values may not affect 3D features of meshes, but manipulations on 3D meshes can directly affect 3D features of the meshes, so it is challenging to generated unnoticeable perturbation for 3D meshes. The detail definition of differentiable rendering is shown in appendix.

2.2 Adversarial Optimization Objective

Our objective is to generate a "3D adversarial mesh" by adding subtle perturbation, such that the machine learning models will make incorrect prediction based on its rendered instance. In the meantime we hope the adversarial meshes remain perceptually realistic for human vision system. Specifically, the optimization objective function is as follows:

$$\mathcal{L}(S^{\mathrm{adv}}) = \mathcal{L}_{\mathrm{adv}}(S^{\mathrm{adv}}, g, y') + \lambda \mathcal{L}_{\mathrm{perceptual}}(S^{adv}) \tag{1}$$

In this equation, $\mathcal{L}_{\mathrm{adv}}$ is the adversarial loss to fool the model g to predict a specified target y' (i.e. $g(I^{\mathrm{adv}}) = y'$), given the rendered image $I^{\mathrm{adv}} = R(S^{\mathrm{adv}}; P, L)$. $\mathcal{L}_{\mathrm{perceptual}}$ is a loss to keep the 3D adversarial meshes perceptually realistic. λ is a hyper-parameter to balance the losses. Given the optimization objective, we try to generate the 3D adversarial mesh S^{adv} by manipulating its shape and texture respectively. We denote this method as *meshAdv*.

We evaluate *meshAdv* on two common tasks: image classification and object detection. We further instantiate $\mathcal{L}_{\mathrm{adv}}$ and $\mathcal{L}_{\mathrm{perceptual}}$ in the next subsections, regarding different tasks and different perturbation sources (vertices or texture).

2.2.1 Adversarial Losses

Classification For a classification model g, the output is the probability distribution of object categories, given an image of the object as the input. We use the cross entropy loss (De Boer et al., 2005) as the adversarial loss for *meshAdv*:

$$\mathcal{L}_{\mathrm{adv}}(S^{\mathrm{adv}}, g, y') = y' \log(g(I^{\mathrm{adv}})) + (1 - y') \log(1 - g(I^{\mathrm{adv}})) \tag{2}$$

Note that image I^{adv} is the rendered image of S^{adv}: $I^{\mathrm{adv}} = R(S^{\mathrm{adv}}; P, L)$.

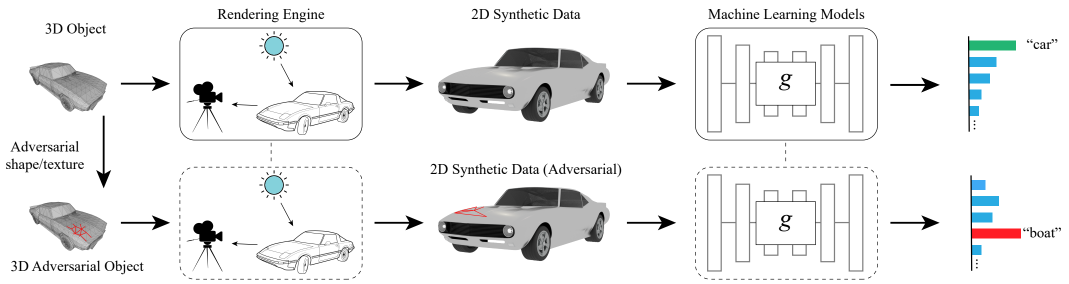


Figure 1: The pipeline of adversarial mesh generation by *meshAdv*.

Object Detection For object detection, we choose the state-of-the-art model, Yolo-v3 (Redmon & Farhadi, 2018), as our target attack model g. It divides the whole input image I into $S \times S$ different grid cells. For each grid cell, Yolo-v3 predicts the locations and label confidence values of B bounding boxes. Then for each bounding box, it generates 5 values (4 for coordinate and 1 for probability of the bounding box) and a probability distribution over N classes. We use the disappearance attack loss (Eykholt et al., 2018) as our adversarial loss:

$$\mathcal{L}_{\text{adv}} = \max_{s \in S^2, b \in B} H(s, b, y', g(I^{\text{adv}})) \tag{3}$$

where $g(I^{\text{adv}})$ is the output of Yolo-v3 by feeding a rendered image $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$. $H(\cdot)$ is a function to extract the probabilities of the bounding box b in grid cell s labeled as target class y' from the output $g(I^{\text{adv}})$.

2.2.2 Perceptual Losses

To keep the "3D adversarial meshes" perceptually realistic, we leverage a smoothing loss similar to the total variation loss (Vogel & Oman, 1996) as our perceptual loss:

$$\mathcal{L}_{\text{perceptual}} = \sum_{i,j} \left\| I^{\text{adv}}_{i+1,j} - I^{\text{adv}}_{i,j} \right\|_2^2 + \left\| I^{\text{adv}}_{i,j+1} - I^{\text{adv}}_{i,j} \right\|_2^2, \tag{4}$$

where i, j are the coordinates of the image I^{adv} rendered with the adversarial meshes S^{adv}.

We apply this smoothing loss when generating perturbations of textures for the adversarial meshes S^{adv}. However, for shape perturbation, manipulation of vertices may introduce unwanted mesh topology change, which is reported in (Kato et al., 2018). In our task, we do not want to directly smooth the vertices, but the displacement of vertices from the original positions instead. Therefore, we extend our 2D smoothing loss to our 3D vertex flows:

$$\mathcal{L}_{\text{perceptual}} = \sum_{\boldsymbol{v}_i^{\text{adv}} \in V^{\text{adv}}} \sum_{\boldsymbol{v}_q^{\text{adv}} \in \mathcal{N}(\boldsymbol{v}_i)} \|\Delta \boldsymbol{v}_i - \Delta \boldsymbol{v}_q\|_2^2, \tag{5}$$

where $\Delta \boldsymbol{v}_i = \boldsymbol{v}_i^{\text{adv}} - \boldsymbol{v}_i$ is the displacement of the vertex $\boldsymbol{v}_i^{\text{adv}}$ from the position $\boldsymbol{v}_i$ in the pristine object, and $\mathcal{N}(v_i)$ denotes vertexes which are on the same face (neighbors) with $\boldsymbol{v}_i$.

3 Transferability of 3D Adversarial Meshes

By optimizing the aformentioned adversarial objective end-to-end, we obtain the "3D adversarial meshes " and fool the network output to our specified target, using the image rendered by the differentiable renderer. However, it is particularly interesting to see whether black-box attack against unknown rendering is possible, and whether we can attack rendering R with low computational cost and transfer such perturbation to R' with high computational cost such as industrial-level rendering. Here we call R' a photorealistic renderer, because multiple-bounce interreflection, occlusion, high quality stratified sampling and reconstruction, complicated illumination models are all present in R' such that the final image is an accurate estimate of real-world physics as being captured by a camera. We analyze two scenarios for transferring in both known and unknown environments.

Transferability to a Photorealistic Renderer in an Known Environment In this scenario, our purpose is to test our 3D adversarial meshes directly under the same rendering configuration (lighting parameters L, camera parameters P), only replacing the the differentiable renderer R with the photorealistic renderer R'. In other words, while $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$ can fool the network g as expected, we would like to see whether $I'^{\text{adv}} = R'(S^{\text{adv}}; P, L)$ can targeted/untargeted g as well.

Transferability to a Photorealistic Renderer in an Unknown Environment In this scenario, we would like to attack a non-differentiable system $g(R'(S; L^{\text{env}}, P^{\text{env}}))$ in a fixed unknown environment with a differentiable render R. We still have the access to the shape S and its mask in the original photorealistic rendering M, and the network g. But the rendering process of R' is not differentiable, and we hope to employ the differentiable renderer to generate the adversarial perturbations and transfer the adversarial behavior to $I'^{\text{adv}} = R'(S^{\text{adv}}; L^{\text{env}}, P^{\text{env}})$ such that I'^{adv} will fool g. To achieve this, we propose the attacking pipeline as follows: 1) Estimate the camera parameters P by optimizing the difference of the two masks $\|R_{\text{mask}}(S; P, L) - M\|^2$, where $R_{\text{mask}}(S; P, L)$ renders the silhouettes of the object S as the object mask; 2) Estimate the lighting parameters L

using the estimated camera parameters P^{est} by optimizing the difference of the two rendered image in the masked region: $\|M \cdot (R(S; P^{\text{est}}, L) - I')\|^2$; Note that R' need not have the same lighting model as in R, but we still can use simple lighting models in R to estimate the complicated lighting L^{env}; 3) Generate the adversarial meshes S^{adv} using our adversarial and perceptual losses with the estimated lighting and camera parameters; here we add randomness to lighting parameters, camera parameters, and the object position to improve the robustness against those variations since we do not have an exact estimate; 4) Test our adversarial object in the photorealistic renderer: $g(R'(S^{\text{adv}}; L^{\text{env}}, P^{\text{env}}))$.

4 Experimental Results

In this section, we first show the 2D examples rendered from "adversarial meshes" generated by *meshAdv* against classifiers under various settings. We then visualize the manipulation flow of vertices to better understand the vulnerable regions for those 3D objects. In addition, we show examples of applying *meshAdv* to object detectors in physically realistic scenes. Finally, We evaluate the transferability for 3D adversarial meshes from the differentiable renderer to a photorealistic non-differentiable renderer under known rendering environments and showcase our solution under an unknown rendering environment.

4.1 Experimental Setup

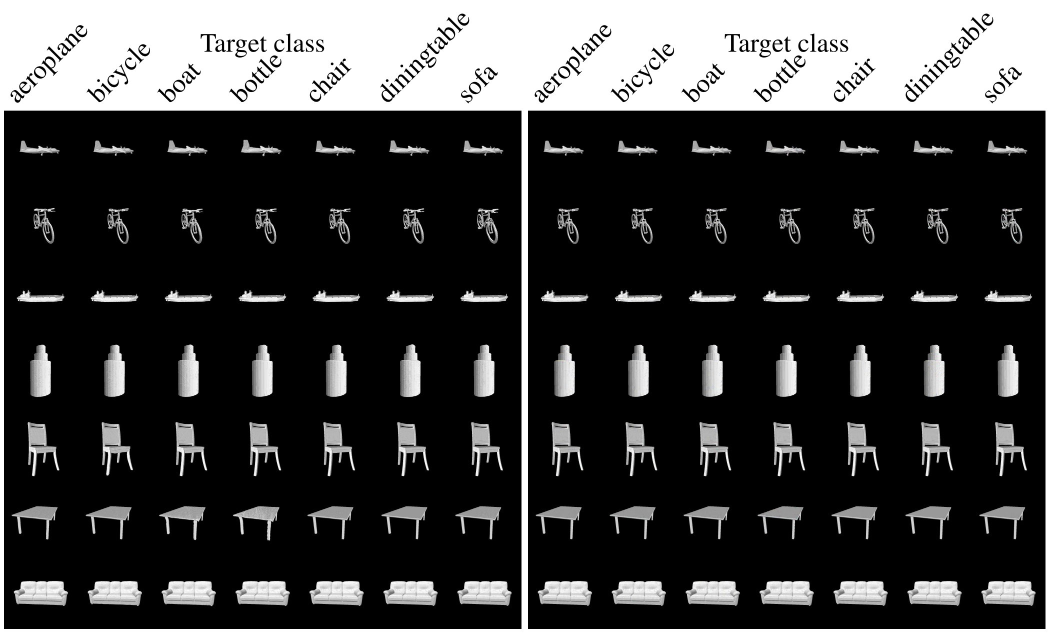
In our experiment, we choose DenseNet (Huang et al., 2017) and Inception-v3 (Szegedy et al., 2016) trained on ImageNet (Deng et al., 2009) as our target attack models for classification, and Yolo-v3 trained on COCO (Lin et al., 2014) for object detection.. We preprocess CAD models in PASCAL3D+ (Xiang et al., 2014) with uniform mesh resampling with MeshLab Cignoni et al. (2008) to increase the triangle density, and use the processed CAD models as 3D objects to attack. Since these 3D objects have constant texture values, for texture perturbation we also start from constant as pristine texture. For the differentiable renderer, we use the off-the-shelf PyTorch implementation (Paszke et al., 2017; Kolotouros, 2018) of the Neural Mesh Renderer(NMR) (Kato et al., 2018) to generate "adversarial meshes". We create a PASCAL3D+ renderings for classification as our evaluation dataset. The details of creation are shown in appendix. We generate a total of 72 samples with 7 different classes of 3D objects. We refer to these data as PASCAL3D+ renderings for later use in this paper. For optimizing the objective, we use Adam (Kingma & Ba, 2014) as our solver. In addition, we select λ using a binary search method, with 5 rounds of search and 1000 iterations for each round. If succeeded, we select the "adversarial meshes" with the lowest $L^{\text{perceptual}}$ during search, for evaluation in the next subsection.

Table 1: Accuracy and attack success rate against different models on pristine data (p) and dversarial examples generated by *meshAdv* with PASCAL3D+ renderings. We show the average distance (mean) and the attack success probabitliy (prob) under different settings.

Adversarial Type	Model	Accuracy	Best Case		Average Case		Worst Case	
			mean	prob	mean	prob	mean	prob
Shape	DenseNet	100%	8.4×10^{-5}	100%	1.8×10^{-4}	100%	3.0×10^{-4}	100%
	Inception-v3	100%	4.76×10^{-5}	100%	1.2×10^{-4}	99.8%	2.3×10^{-4}	98.6%
Texture	DenseNet	100%	3.8×10^{-3}	100%	1.1×10^{-2}	99.8%	2.6×10^{-2}	98.6%
	Inception-v3	100%	3.7×10^{-3}	100%	1.3×10^{-2}	100%	3.2×10^{-2}	100%

4.2 *MeshAdv* on Classification

In this section, we evaluate quantitative and qualitative performances of *meshAdv* for classifiers. For each sample in our PASCAL3D+ renderings, we try to targeted-attack it into the other 6 categories. Next, for each type (shape and texture) and each model (DenseNet and Inception-v3), we split the results into three different cases similar to Carlini & Wagner (2017): *Best Case* means we perform the attack against all incorrect labels and report on the target class that is easiest to attack. *Average Case* means we do the same except we report the performance on all of those targets. Similarly, *Worst case* means that we report on the target class that is hardest to attack. Table 1 shows the attack success rates and their corresponding smoothing loss for adversarial vertex and $L2$ distance

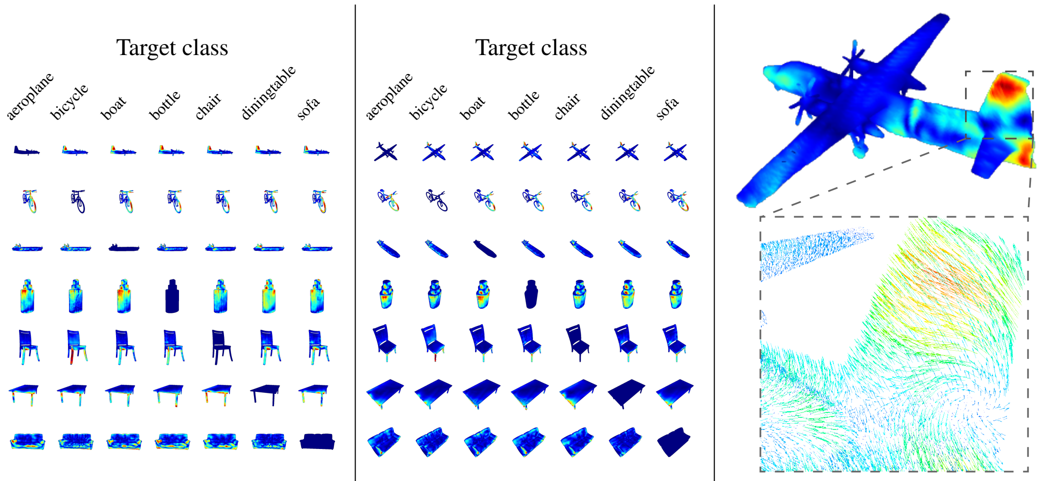


(a) Perturbations on shape (b) Perturbations on texture

Figure 2: Benign images (diagonal) and corresponding adversarial examples generated by *meshAdv* on PASCAL3D+ renderings tested on Inception-v3. Adversarial targeted are shown on the top. We show both perturbations on (a) shape and (b) texture.

for adversarial texture on the PASCAL3D+ images rendered with "adversarial meshes" generated by *meshAdv*. The results show that *meshAdv* can achieve almost 100% attack success rate for both attacking types.

Figure 2 shows our PASCAL3D+ renderings of the "3D adversarial meshes" for Inception-v3, after manipulating the vertices and textures respectively, while the diagonal is the images rendered with the pristine objects. The target class of each adversarial image is shown at the top, and please see appendix to see the results for DenseNet, which are similar. Note that for each class, we randomly select one sample to show in the image, i.e. these images are not manually curated. It is also worth noting that the perturbations on object shape or texture, generated by our *meshAdv*, are barely noticeable to humans, but have the ability to mislead the classifiers. To help assess the vertex displacement, we discuss the visualization in the following subsection.



(a) Rendered view (b) Canonical view (c) Flow visualization of an 3D adversarial mesh targeting "bicycle"

Figure 3: Heatmap visualization of vertex flows on Inception-v3.

Visualizing shape deformation In order to better understand the vulnerable regions of the 3D objects, in Figure 3, we convert the vertex manipulation flow magnitude to heatmap visualization, for the shapes corresponding to Figure 2(a). We adopt two viewpoints in this figure: the rendered view which is the same as the one used for rendering the images, and the canonical view which is achieved by fixing camera parameters. We observe that the regions that are close to the camera with large curvature, such as edges, are more vulnerable. This is reasonable, since vertex displacement in those regions will bring significant change to normals, thus affecting the shading from the light sources and causing the screen pixel value change more drastically.

Since the heatmap only shows the magnitude of the vertex displacement, and we would also like to observe their directions as well as their magnitude. Figure 3c shows a close-up 3D quiver plot of the vertex flows in the vertical stabilizer region of an aeroplane. In this example, the perturbed aeroplane object is classified to "bicycle" in its rendering. From this figure, we observe that the adjacent vertices flow towards the similar direction, which illustrates the effect of our smoothing loss operated on vertex flows.

Human Perceptual Study The detail description of our human study settings is shown in appendix. In total, we collect 3820 annotations from 49 participants. $99.29 \pm 1.96\%$ of trials were classified correctly, which shows that the our adversarial is highly unnoticeable.

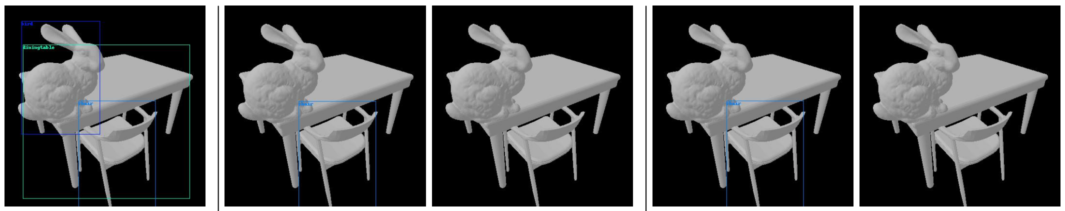

(a) Benign (b) Table | Shape (c) All | Shape (d) Table | Texture (e) All | Texture

Figure 4: 3D adversarial meshes generated by *meshAdv* in a synthetic scene. (a) represents the benign rendered image and (b)-(e) represent the rendered images from "adversarial object" by manipulating the shape or texture. We use the format "adversarial target | adversarial type" to denote the object aiming to hide and the type of perturbation.

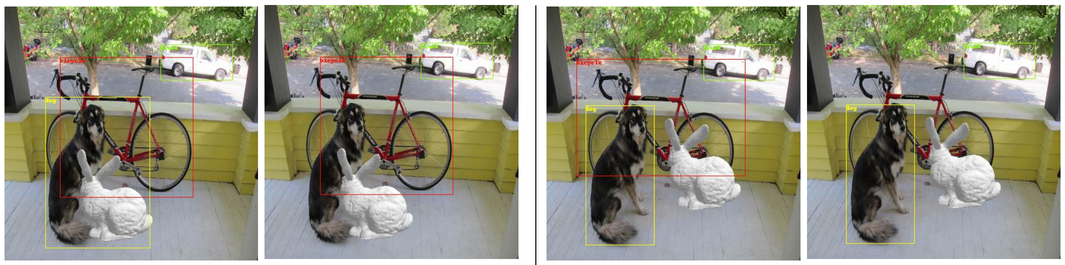

(a) S | - (b) S^{adv} | Dog (c) S | - (d) S^{adv} | Bicycle

Figure 5: 3D adversarial meshes generated by *meshAdv* in an outdoor simulated scene. (a) and (c) are the corresponding rendered images with "pristine object" as control experiment, while (b) and (d) contain "adversarial meshes" by manipulating the shape. We use the format " S/S^{adv} | target" to denote the benign/adversarial 3D meshes and the target to hide from the detector.

4.3 *MeshAdv* on Object Detection

For object detection, we use Yolo-v3 as our detector. In this experiment, we showcase two scenarios to demonstrate that with our *meshAdv*, we can attack the object detector with ease.

Indoor Scene The indoor scene is purely synthetic. We compose the scene manually with a desk and a chair to simulate an indoor environment, and place in the scene a single directional light with low ambient light. We then put the Stanford Bunny (Turk & Levoy, 1994) object onto the desk, and show that by manipulating either the shape or the texture of the adversarial mesh, we can achieve the goal of removing the table detection or removing all detections, while keep the perturbations unnoticeable, as is shown in Figure 4.

Outdoor Scene For the outdoor scene, we take a different approach: given a real photo of the outdoor scene, we first estimate the parameters of a sky lighting model (Hosek & Wilkie, 2012), using the API provided by Hold-Geoffroy et al., and then estimate a directional light and the ambient light. With this light estimation, our adversarial mesh will be realistic when being rendered and put onto the real image. In the real image, we select dog and bicycle as our target objects. Different from adversarial goals in the synthetic indoor scene, we aims to remove the target objects respectively to increase the diversity of adversarial targets. In order to removing them respectively, we put our "adversarial mesh" around the bounding box of the target object. We also successfully fool the network using *meshAdv* with barely noticeable perturbations, and the results are shown in Figure 5.

Table 2: Untargeted attack success rate against Mitsuba by transferring adversarial meshes generated by attacking a differentible rendering targeting on different classes.

model/target	aeroplane	bicycle	boat	bottle	chair	diningtable	sofa	average
DenseNet	65.2%	69.1%	66.7%	63.0%	37.08%	70.3%	47.9%	59.8%
Inception-v3	67.1%	83.3%	39.6%	76.9%	32.1%	75.0%	52.3%	60.9%

4.4 Transferability to a Photorealistic Renderer in Known Environment

We directly render our "adversarial meshes" generated from the section 4.2 using a photorealistic rendering called Mitsuba (Jakob, 2010), with the same lighting and viewpoints. We then evaluate the targeted/untargeted attack transferability by feeding the newly rendered images. The results are shown in Table 2. We observe that the the "adversarial meshes" can be easily transferred to Mitsuba rendering without any modifications om untargeted attack. The confusion matrices of the transferability related to targeted attack are shown in appendix. The diagonal is the accuracy of the classifier on the Mitsuba-rendered images of pristine meshes. $\mathrm{Cell}(i, j)$ represents the attack success rate of the "adversarial meshes" labeled as i and be attacked as target label j on Mitsuba rendering. Different from the high transferability of untargeted attack, the transferability is pretty low except for some similar objects.

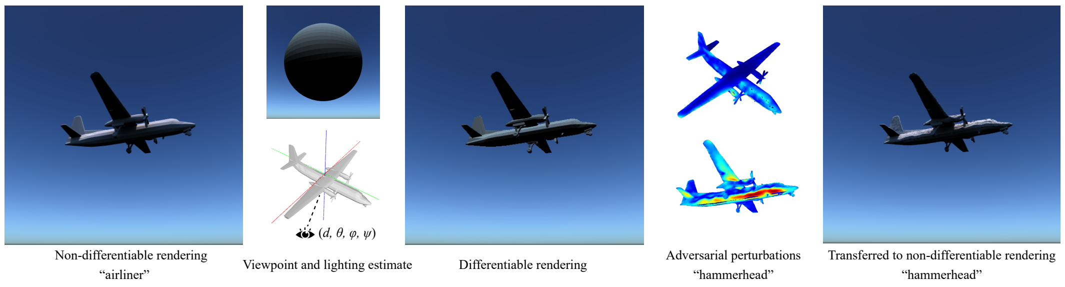
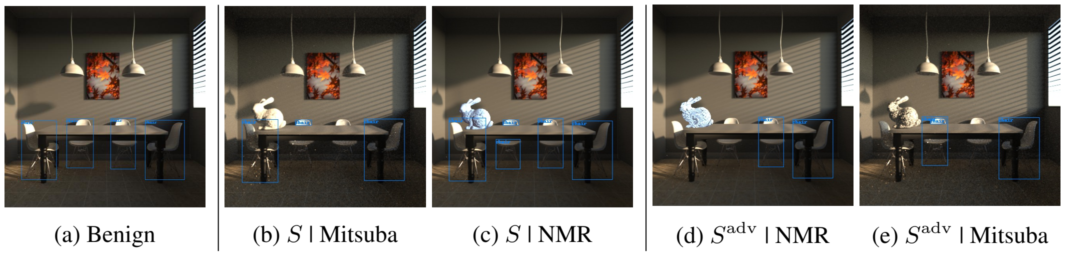

Non-differentiable rendering
"airliner"
$(d, \theta, \varphi, \psi)$
Viewpoint and lighting estimate
Differentiable rendering
Adversarial perturbations
"hammerhead"
Transferred to non-differentiable rendering
"hammerhead"

Figure 6: 3D adversarial mesh for classification under unknown environment. We estimate the viewpoint and lighting parameters with the differentiable renderer, and then use the estimated environment to perform targeted attack on classification using the differentiable renderer, and transfer to the non-differentiable renderer.

4.5 Transferability to a Photorealistic Renderer in Unknown Environment

We compose two scenes, for classification and object detection respectively. Following the estimation method in an unknown environment in section 3, we first optimize the camera parameters P^{est} using the Adam optimizer (Kingma & Ba, 2014), then estimate the lighting L^{est} using 5 directional lights and an ambient light. Then we use NMR to manipulate the shape S^{adv} in the scene until the image I^{adv} rendered by NMR sucessfully targeted-attack the classifier or the object detector g with a high confidence. During this process, we add small random perturbations to the camera P^{est},



(a) Benign (b) S | Mitsuba (c) S | NMR (d) S^{adv} | NMR (e) S^{adv} | Mitsuba

Figure 7: 3D adversarial meshes against object detectors in photorealistic scenes. we use "S | renderer" to denote the whether the added object is adversarially optimized and the rendering engine that we aim to transfer the perturbation to attack against.

lighting L^{est} and the 3D object position such that our generated "adversarial meshes" S^{adv} will be more robust. After successfully generating S^{adv}, we re-render the original scene with S^{adv}, and test the rendering $I'^{\mathrm{adv}} = R'(S^{\mathrm{adv}}, P^{\mathrm{env}}, L^{\mathrm{env}})$ on the model g.

Transferability for Classification We place an aeroplane object from PASCAL3D+ and put it in an outdoor sky light, and render it using Mitsuba. As is shown in figure 9, we successfully attacked the classification system to output the our target "hammerhead" by replacing the pristine object with our "adversarial meshes". Note that even we did not have a very accurate lighting estimate, we still achieve the transferability by adding perturbations in the lighting parameters.

Transferability for Object Detection For object detection, we modified a scene from Bitterli (2016), and placed the Stanford Bunny object into the scene. Similarly, without an accurate lighting estimate, our "adversarial meshes" successfully removed the detection prediction for our target chair (the leftmost one).

5 Conclusion

In this paper, we proposed *meshAdv* to generate "3D adversarial meshes" by manipulating the texture or the shape information. These "3D adversarial meshes" can be rendered to 2D domain to mislead different machine learning models. We provide in-depth analysis for the vulnerable regions for 3D objects based on the visualization of the vertex flows and we also analyze the transferability for 3D adversarial meshes among different renderings systems and show that untargeted attack can achieve high transferability which is not true for targeted attack, which provides us better understand adversarial behaviors in practice and motive potential defense.

References

Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015.

Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 39–57, 2017. doi: 10.1109/SP.2017.49. URL `https://doi.org/10.1109/SP.2017.49`.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing 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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV)*, 2015.

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pp. 129–136, 2008.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM, 2008.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

Ltsc Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael L Seltzer, Geoffrey Zweig, Xiaodong He, Jason D Williams, et al. Recent advances in deep learning for speech research at microsoft. In *ICASSP*, volume 26, pp. 64, 2013.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1, 2017.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Physical adversarial examples for object detectors. *arXiv preprint arXiv:1807.07769*, 2018.

Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3d morphable model regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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

Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding realworld indoor scenes with synthetic data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4077–4085, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017.

Lukas Hosek and Alexander Wilkie. An analytic model for full spectral sky-dome radiance. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4), July 2012. To appear.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, pp. 3, 2017.

David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pp. 133–142, New York, NY, USA, 1986. ACM. ISBN 0-89791-196-2. doi: 10.1145/15922.15901. URL `http://doi.acm.org/10.1145/15922.15901`.

Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pp. 143–150, New York, NY, USA, 1986. ACM. ISBN 0-89791-196-2. doi: 10.1145/15922.15902. URL `http://doi.acm.org/10.1145/15922.15902`.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikos Kolotouros. Pytorch implememtation of the neural mesh renderer. `https://github.com/daniilidis-group/neural_renderer`, 2018. Accessed: 2018-09-10.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Adversarial geometry and lighting using a differentiable renderer. *CoRR*, abs/1808.02651, 2018.

Matthew M. Loper and Michael J. Black. Opendr: An approximate differentiable renderer. In *Computer Vision – ECCV 2014*, pp. 154–169, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10584-0.

Francisco Massa, Bryan Russell, and Mathieu Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

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*, pp. 2574–2582, 2016.

Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems*. 2018.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pp. 497–500. ACM, 2001. ISBN 1-58113-374-X. doi: 10.1145/383259.383317.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pp. 102–118. Springer International Publishing, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pp. 311–318, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. doi: 10.1145/192161.192241. URL `http://doi.acm.org/10.1145/192161.192241`.

Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017.

Curtis R Vogel and Mary E Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996.

Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pp. 75–82. IEEE, 2014.

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*, 2018a.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018b. URL `https://openreview.net/forum?id=HyydRMZC-`.

Dawei Yang and Jia Deng. Shape from shading through shape evolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

A. Khosla F. Yu L. Zhang X. Tang J. Xiao Z. Wu, S. Song. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015.

Xiaohui Zeng, Chenxi Liu, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi Keung Tang, and Alan L Yuille. Adversarial attacks beyond the image space. *arXiv preprint arXiv:1711.07183*, 2017.

Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

APPENDIX

A RELATED WORK

Differentiable 3D rendering system Our method integrates gradient-based optimization with a differentiable 3D rendering engine integrated into an end-to-end pipeline. There are different 3D rendering Barron & Malik use a spherical-harmonics-lighting-based differentiable renderer (Ramamoorthi & Hanrahan, 2001) to jointly estimate shape, reflectance and illumination from shading by optimizing to satisfy the rendering equation. Kato et al. propose the Neural Mesh Renderer for neural networks and perform single-image 3D mesh reconstruction and gradient-based 3D mesh editing with the renderer. Genova et al. integrate a differentiable renderer during training, to regress 3D morphable model parameters from image pixels. Mordvintsev et al. show that through differentiable rendering, they can perform texture optimization and style transfer directly in screen space to achieve better visual quality and 3D properties. These gradient-based optimization methods with rendering integrated pipelines, along with our work, largely attribute to the readily accessible differentiable renderers such as OpenDR (Loper & Black, 2014), differentiable mesh renderers designed for neural networks (Kato et al., 2018; Genova et al., 2018), RenderNet (Nguyen-Phuoc et al., 2018), and the irradiance renderer (Ramamoorthi & Hanrahan, 2001; Barron & Malik, 2015). We exploit differentiable renderering techniques with a different optimization target: we try to minimize our modification to 3D contents, while deceiving a machine learning model into misclassification or misdetection for objects in rendered images.

Adversarial attacks Adversarial examples have been heavily explored in 2D domains (Szegedy et al., 2013; Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016). Physical adversarial examples are studied by (Kurakin et al., 2016; Evtimov et al., 2017; Athalye & Sutskever, 2017). However, they focus on manipulating the paints color of the objects and do not focus on the 3D object itself. In this work, we aim to explore the adversarial 3D mesh itself. Zeng et al. (2017) perturbed the physical parameters (normal, illumination and material) for untargeted attacks against 3D shape classification and a visual question answering system. However, they represent the shape with pixelwise normal map, which still operates on 2D space, and such normal map may not be physically plausible. A concurrent work (Liu et al., 2018) proposes to manipulate lighting and geometry to attack 3D rendering engine. However, there are several major differences comparing with our work: 1) **Magnitude of perturbation**. The perturbation in Liu et al. (2018) such as lighting change is visible, while low magnitude perturbation is the most important part in adversarial behaviors. In our proposed attacks, we explicitly constraint the perturbation to be of small magnitude, and we conduct human subject experiments to confirm that the perturbation is unnoticeable. 2) **Targeted attack.** Based on the objective function and results of the experiments, Liu et al. (2018) can only mislead objects from one category to other close categories such as jaguar and elephant. In our work, we explicitly force the object from each class to be targeted attacked into all the rest of classes with almost 100% attack success rate. 3) **Rendering Engine.** We perform attacks based on the state of the art rendering engine (Kolotouros, 2018) which makes our attacks, while Liu et al. (2018) built a customized rendering engine and it is hard to tell whether such vulnerabilities come from the customized rendering or the manipulated object. (4). **Realistic attacks.** Manipulating lighting is less realistic in open environments. Compared with their attacks on lighting and shape, we proposed to manipulate shape and texture of meshes which are easier to conduct in practice. In addition, we evaluate our attacks in random physically realistic scenes to demonstrate the robustness of attacks under various physical conditions. (5). **Victim learning models.** We attack both classifier and object detector, which is widely used in safety-sensitive applications such as autonomous driving, while Liu et al. (2018) only attacks classifiers. (6). **Understandings.** We provide in-depth analysis for the 3D adversarial examples, such as their vulnerable regions, to help build better understanding.

B EXPERIMENT SETTINGS

Creation of PASCAL3D+ renderings data We state the creation of our PASCAL3D+ renderings for classification. First, we use NMR to generate synthetic renderings using the objects in PASCAL3D+ in different settings such as viewpoints and lighting (intensity and direction). Then, we create a table mapping the object classes in PASCAL3D+ to the corresponding classes in the ImageNet. Next, we feed the synthetic renderings to DenseNet and Inception-v3 and filter out the samples that are misclassified by either network, which means both models have 100% prediction accuracy on our PASCAL3D+ renderings.

C DIFFERENTIABLE RENDERING FORMULATION

A physically-based rendering engine R computes a 2D image $I = R(S; P, L)$ with camera parameters P, 3D mesh S and illumination parameters L by approximating the real world physics, e.g. the rendering equation (Kajiya, 1986; Immel et al., 1986). A differentiable rendering engine makes such computation differentiable w.r.t. the input S, P, L by making assumptions on lighting models and surface reflectance, and simplifying the ray-casting process.

Following common practice, we use 3D triangular meshes for shape representation, Lambertian surface for surface modeling, directional lighting with a uniform ambientfor illumination, and ignore interreflection and occlusion. Under these assumptions, if a triangular mesh S has vertices V, faces F and textures T, given camera parameters P, 3D mesh S and light sources L, the 2D image produced by the differentiable renderer can be derived as

$$I = \text{rasterize}(P, T \cdot \text{shading}(L, \text{normal}(V, F))), \tag{6}$$

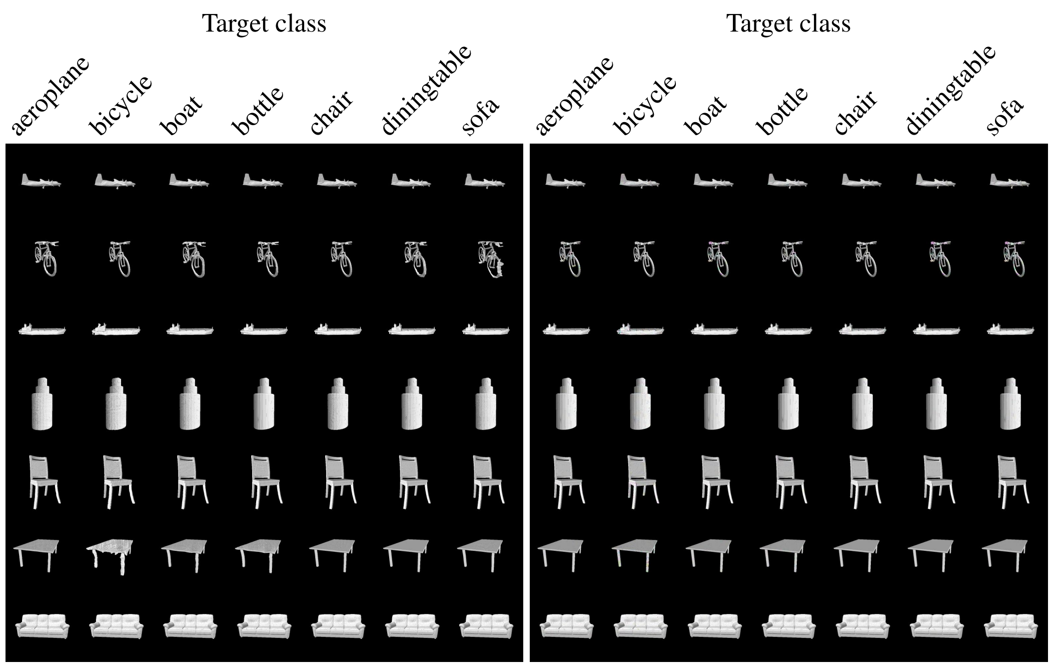
where "normal" computes the normal directions of surfaces, "shading" computes the shading colors for each face, and "rasterize" maps the colors of faces onto the screen space by raycasting. The above functions are all differentiable with respect to the inputs, such that the whole rendering process is differentiable.

We further explain the details regarding 3D object $S = (V, F, T)$, illumination L and camera parameters P. For a 3D object S in 3D triangular mesh representation, let V be the set of its n vertices $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n \in \mathbb{R}^3\}$ in 3D space, and F be the indices of its m faces $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_m \in \mathbb{N}^3\}$. For textures, traditionally, they are represented by mapping to 2D images with mesh surface parameterization. For simplicity, here we attach to each triangular face a single color as its texture: $T = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_m \in {\mathbb{R}^+}^3\}$.

For illumination, we use directional light sources plus an ambient light. The lighting directions are denoted $L_{\text{dir}} = \{\boldsymbol{l}_1^d, \boldsymbol{l}_2^d, \cdots \in \mathbb{R}^3\}$, where $\boldsymbol{l}_i^d, i \in \mathbb{N}$ are unit vectors. Similarly, the lighting colors are denoted $L_{\text{color}} = \{\boldsymbol{l}_1^c, \boldsymbol{l}_2^c, \cdots\}$ for directional light sources and $\boldsymbol{a}$ for the ambient light, with $\boldsymbol{l}_i, \boldsymbol{a} \in {\mathbb{R}^+}^3$ in RGB color space.

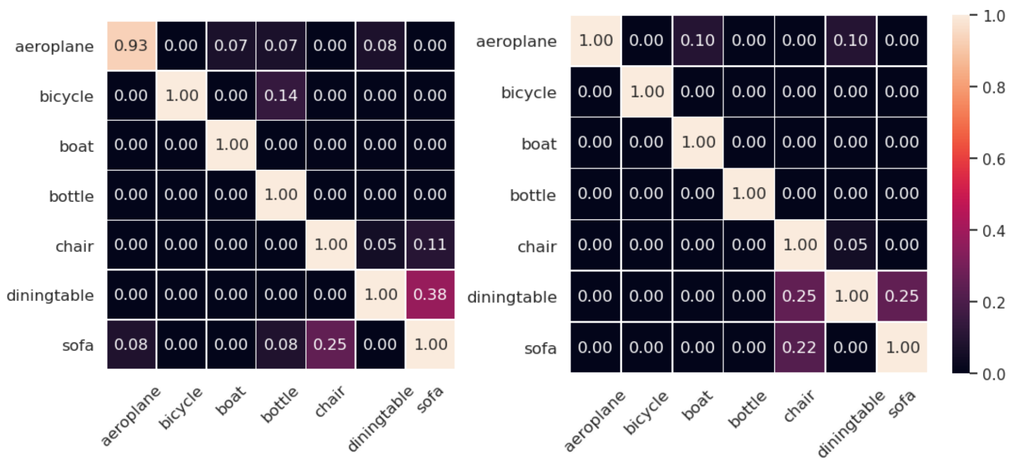
We put the object $S = (V, F, T)$ at the origin $(0, 0, 0)$, and set up our perspective camera following a common practice: the camera viewpoint is described by a quadraple $P = (d, \theta, \phi, \psi)$, where d is the distance of the camera to the origin, and θ, ϕ, ψ are azimuth, elevation and tilt angles respectively. Note that here we assume the camera intrinsics are fixed and we only need gradients for the extrinsic parameters P.

Human Perceptual Study We conduct a user study on Amazon Mechanical Turk (AMT) in order to quantify the realism of the adversarial meshes generated by *meshAdv*. We uploaded the adversarial images on which DenseNet and Inception-v3 misclassify the object. Participants were asked to classify those adversarial images to one of the two classes (the ground-truth class and the target class).



(a) Shape (b) Texture

Figure 8: Benign images and corresponding adversarial examples generated by *meshAdv* on PASCAL3D+ on DenseNet. (a) presents the "adversarial meshes" by manipulating shape while (b) by manipulating texture change.



(a) DenseNet (b) Inception-v3

Figure 9: Target attack transferability to mitsuba. (i,j) represents the attack success rate of the "adversarial meshes" labeled as groundtruth i and attacked as target j on Mitsuba rendering.