MeshAdv: Adversarial Meshes for Visual Recognition

Chaowei Xiao [*1] Dawei Yang [*1,2] Jia Deng [2] Mingyan Liu [1] Bo Li [3]
[1] University of Michigan, Ann Arbor
[2] Princeton University
[3] UIUC

Abstract

Highly expressive models such as deep neural networks (DNNs) have been widely applied to various applications. However, recent studies show that DNNs are vulnerable to adversarial examples, which are carefully crafted inputs aiming to mislead the predictions. Currently, the majority of these studies have focused on perturbation added to image pixels, while such manipulation is not physically realistic. Some works have tried to overcome this limitation by attaching printable 2D patches or painting patterns onto surfaces, but can be potentially defended because 3D shape features are intact. In this paper, we propose meshAdv to generate "adversarial 3D meshes" from objects that have rich shape features but minimal textural variation. To manipulate the shape or texture of the objects, we make use of a differentiable renderer to compute accurate shading on the shape and propagate the gradient. Extensive experiments show that the generated 3D meshes are effective in attacking both classifiers and object detectors. We evaluate the attack under different viewpoints. In addition, we design a pipeline to perform black-box attack on a photorealistic renderer with unknown rendering parameters.

1. Introduction

Despite the increasing successes in various domains [10, 13, 19, 44], deep neural networks (DNNs) are found vulnerable to adversarial examples: a deliberate perturbation of small magnitude on the input can make a network output incorrect predictions. Such adversarial examples have been widely studied in 2D domain [5, 17, 35, 38, 47, 53–55], but the attack generated by directly manipulating pixels can be defended by securing the camera, so that the generated images are not realizable in practice. To overcome this issue, there has been significant prior progress on generating physically possible adversarial examples [1, 4, 14, 27] by altering the texture of a 3D surface, *i.e.* applying adversarial printable 2D patches or painting patterns. Such attacks, however, are less suitable for textureless objects, because adding texture to an otherwise textureless surface may increase the chance of being detected and defended.

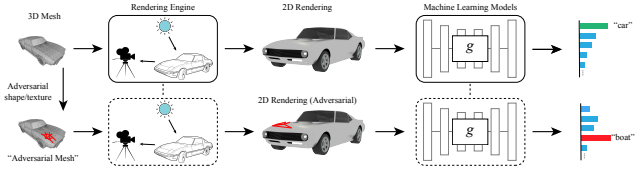


Figure 1: The pipeline of "adversarial mesh" generation by *meshAdv*.

In this work, we explore a new avenue of attack where we generate physically possible adversarial examples by altering 3D shape. We explore 3D objects that have rich shape features but minimal texture variation, and show that we can still fulfill the adversarial goal by perturbing the shape of those 3D objects, while the same method can still be applied to textures. Specifically, we propose *meshAdv* to generate adversarial meshes with negligible perturbations. We leverage a physically based differentiable renderer [24] to accurately render the mesh under certain camera and lighting parameters. A deep network then outputs a prediction given the rendered image as input. Since this whole process is differentiable, gradients can be propagated from the network prediction back to the shape or texture of the mesh. Therefore, we can use gradient based optimization to generate shape based or texture based perturbation by applying losses on the network output. The entire pipeline is shown in Figure 1.

Even though we are only manipulating physical properties (shape and texture) of a 3D object, we can always fool state of the art DNNs (see Section 6.2). Specifically, we show that for any fixed rendering conditions (*i.e.* lighting and camera parameters), state of the art object classifiers (DenseNet [22] and Inception-v3 [48]) and detector (Yolo-

[*]Alphabetical ordering; The first two authors contributed equally.

v3 [42]) can be consistently tricked by slightly perturbing the shape and texture of 3D objects. We further show that by using multiple views optimization, the attack success rate of "adversarial meshes" increases under various viewpoints (see Table 2). In addition, we conduct user studies to show that the generated perturbation are negligible to human perception.

Since the perturbation on meshes is adversarially optimized with the help of a differentiable renderer, a natural question to ask is whether a similar method can be applied in practice when the rendering operation is not differentiable. We try to answer this question by proposing a pipeline to perform black-box attack on a photorealistic renderer (with non-differentiable rendering operation) under unknown rendering parameters. We show that via estimating the rendering parameters and improving the robustness of perturbation, our generated "adversarial meshes" can attack on a photorealistic renderer.

Additionally, we visualize our shape based perturbation to show possible vulnerable regions for meshes. This can be beneficial when we hope to improve the robustness (against shape deformation) of machine learning models that are trained on 3D meshes [7, 45, 57] for different tasks such as view point estimation [46], indoor scene understanding [18, 34, 45, 59] and so on [8, 33, 43, 50, 56].

To summarize, our *contributions* are listed below: 1) We propose an end-to-end optimization based method *meshAdv* to generate 3D "adversarial meshes" with negligible perturbations, and show that it is effective in attacking different machine learning tasks; 2) We demonstrate the effectiveness of our method on a black-box non-differentiable renderer with unknown parameters; 3) We provide insights into vulnerable regions of a mesh via visualizing the flow of shape based perturbations; 4) We conduct user studies to show that our 3D perturbation is subtle enough and will not affect user recognition.

2. Related Work

Adversarial Attacks Adversarial examples have been heavily explored in 2D domains [17, 35, 38, 47, 54, 55], but directly manipulation of image pixels requires access to cameras. To avoid this, physical adversarial examples studied in [14, 27] show impressive robust adversarial examples under camera transformations. However, the perturbations are textured based and may not be applied to arbitrary 3D shapes.

Meanwhile, Athalye et al. [1] further advance texture based adversarial examples by enhancing the robustness against color transformations, and show that the generated textures for a turtle and a baseball that can make the them fool a classifier under various different viewpoints. In this exciting work, the 3D objects serve as a surface to carry information-rich and robust textures that can fool classifiers. In our work, we also focus on perturbation on 3D objects, but we explicitly suppress the effect of textures by starting from 3D objects [52] that have constant reflectance. Even with constant reflectance, those 3D objects such as airplanes, bicycles, are easily recognizable due to their distinctive 3D shape features. In this way, we highlight the importance of these shape features of objects in adversarial attacks.

Beyond perturbations in texture form, Zeng et al. [58] perturbed the physical parameters (normal, illumination and material) for untargeted attacks against 3D shape classification and a VQA system. However, for the differentiable renderer, they assume that the camera parameters are known beforehand and then perturb 2D normal maps under the fixed projection. This may limit the manipulation space and may also produce implausible shapes. For the non-differentiable renderer in their work, they have to use derivative-free optimization for attacks. In comparison, our method can generate plausible shapes directly in mesh representation using gradient based optimization methods.

A concurrent work [30] proposes to manipulate lighting and geometry to perform attacks. However, there are several differences compared to our work: 1) *Magnitude of perturbation.* The perturbation in [30] such as lighting change is visible, while we achieve almost unnoticeable perturbation which is important in adversarial behaviors. 2) *Targeted attack.* Based on the objective function in [30] and experimental results, the adversarial targets seem close to each other, such as jaguar and elephant. In our work, we explicitly force the object from each class to be targeted-attacked into all other classes with almost 100% attack success rate. 3) *Renderers.* We perform attacks based on the state-of-the-art open-source differentiable renderer [26], which makes our attacks more accessible and reproducible, while in [30] a customized renderer is applied and it is difficult to tell whether such vulnerabilities come from the customized renderer or the manipulated object. 4) *Realistic attacks.* Manipulating lighting is less realistic in open environments. Compared with their attacks on lighting and shape, we propose to manipulate shape and texture of meshes which are easier to conduct in practice. 5) *Victim learning models.* We attack both classifiers and object detectors, which is widely used in safety-sensitive applications such as autonomous driving, while they only attack classifiers.

Differentiable Renderers Besides adversarial attacks, differentiable renderers have been used in many other tasks as well, including inverse rendering [2, 16], 3D morphable face reconstruction [16], texture optimization [36] and so on [28]. In these tasks, gradient based optimization can be realized due to readily available differentiable renderers [16, 24, 28, 31, 37, 41]. We also used a differentiable renderer called Neural Mesh Renderer [24], which is fast and can be integrated into deep neural networks effortlessly.

Watermarking for Meshes While mesh watermarking is also achieved by manipulating the meshes in a subtle way, the goal is different from ours: it is to hide secret data in the geometry by satisfying strict properties of vertices and edges [6, 40]; our task is to perturb the mesh as long as the rendered image can fool a learning model while keeping the mesh perceptual realistic. On the other hand, the challenges in developing 3D mesh watermarking helps to emphasize the challenges for our attack given the difficulties of generating perturbation in 3D domains.

3. Problem Definition and Challenges

In 2D domain, let g be a machine learning model trained to map a 2D image I to its category label y. For g, an adversarial attacker targets to generate an adversarial image I^{adv} such that $g(I^{\text{adv}}) \neq y$ (untargeted attack) or $g(I^{\text{adv}}) = y'$ (targeted attack), where y is the groundtruth label and y' is our specified malicious target label.

Unlike adversarial attacks in 2D space, here the image I is a rendered result of a 3D object S: $I = R(S; P, L)$, computed by a physically based renderer R with camera parameters P and illumination parameters L. In other words, it is not allowed to directly operate the pixel values of I, and one has to manipulate the 3D object S to generate S^{adv} such that the rendered image of it will fool g to make incorrect predictions: $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$.

Achieving the above goals is non-trivial due to the following challenges: 1) **Manipulation space**: When rendering 3D contents, shape, texture and illumination are entangled together to generate the pixel values in a 2D image, so image pixels are no longer independent with each other. This means the manipulation space can be largely reduced due to image parameterization. 2) **Constraints in 3D**: 3D constraints such as physically possible shape geometry and texture are not directly reflected on 2D [58]. Human perception of an object are in 3D or 2.5D [32], and perturbation of shape or texture on 3D objects may directly affect human perception of them. This means it can be challenging to generate unnoticeable perturbations on 3D meshes.

4. Methodology

Here we assume the renderer R is known (*i.e.* white box) and differentiable to the input 3D object S in mesh representation. To make a renderer R differentiable, we have to make several assumptions regarding object material, lighting models, interreflection *etc*. Please refer to supplementary material for more details on differentiable rendering and mesh representation. With a differentiable renderer, we can use gradient-based optimization algorithms to generate the mesh perturbations in an end-to-end manner, and we denote this method *meshAdv*.

4.1. Optimization Objective

We optimize the following objective loss function with respect to S^{adv}, given model g and target label y':

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

In this equation, $\mathcal{L}_{\text{adv}}$ is the adversarial loss to fool the model g into predicting a specified target y' (*i.e.* $g(I^{\text{adv}}) = y'$), given the rendered image $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$ as input. $\mathcal{L}_{\text{perceptual}}$ is the loss to keep the "adversarial mesh" perceptually realistic. λ is a balancing hyper-parameter.

We further instantiate $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{perceptual}}$ in the next subsections, regarding different tasks (classification or object detection) and perturbation types (shape or texture).

4.1.1 Adversarial Loss

Classification For a classification model g, the output is usually the probability distribution of object categories, given an image of the object as the input. We use the cross entropy loss [11] as the adversarial loss for *meshAdv*:

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

where $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$, and y' is one-hot representation of the target label.

Object Detection For object detection, we choose a state-of-the-art model Yolo-v3 [42] as our victim model. It divides the input image I into $Z \times Z$ different grid cells. For each grid cell, Yolo-v3 predicts the locations and label confidence values of B bounding boxes. For each bounding box, it generates 5 values (4 for the coordinates and 1 for the objectness score) and a probability distribution over N classes. Here the adversary's goal is to make the victim object disappear from the object detector, called *disappearance attack*. So we use the disappearance attack loss [15] as our adversarial loss for Yolo-v3:

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

where $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$, and $H(\cdot)$ is a function to represent the probabilities of label y' for bounding box b in the grid cell z, given I^{adv} as input of model g.

4.1.2 Perceptual Loss

To keep the "adversarial mesh" perceptually realistic, we leverage a Laplacian loss similar to the total variation loss [51] as our perceptual loss:

$$\mathcal{L}_{\text{perceptual}}(S^{\text{adv}}) = \sum_i \sum_{q \in \mathcal{N}(i)} \|I_i^{\text{adv}} - I_q^{\text{adv}}\|_2^2, \quad (4)$$

where I_i is the RGB vector of the i-th pixel in the image $I^{\text{adv}} = R(S^{\text{adv}}; P, L)$, and $\mathcal{N}(i)$ is the 4-connected neighbors of pixel i.

We apply this smoothing loss to the image when generating texture based perturbation for S^{adv}. However, for shape based perturbation, manipulation of vertices may introduce unwanted mesh topology change, as reported in [24]. Therefore, instead of using Eq. (4), we perform smoothing on the displacement of vertices such that neighboring vertices will have similar displacement flow. We achieve this by extending the smoothing loss to 3D vertex flow, in the form of a Laplacian loss:

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

where $\Delta \boldsymbol{v}_i = \boldsymbol{v}_i^{\text{adv}} - \boldsymbol{v}_i$ is the displacement of the perturbed vertex $\boldsymbol{v}_i^{\text{adv}}$ from its original position $\boldsymbol{v}_i$ in the pristine mesh, and $\mathcal{N}(\boldsymbol{v}_i)$ denotes connected neighboring vertices of $\boldsymbol{v}_i$ defined by mesh triangles.

5. Transferability to Black-Box Renderers

Our *meshAdv* aims to white-box-attack the system $g(R(S; P, L))$ by optimizing S end-to-end since R is differentiable. However, we hope to examine the potential of *meshAdv* for 3D objects in practice, where the actural renderer may be unavailable.

We formulate this as a black-box attack against a non-differentiable renderer R' under unknown rendering parameters P^*, L^*, *i.e.* we have limited access to R' but we still want to generate S^{adv} such that $R'(S^{\text{adv}}, P^*, L^*)$ fools the model g. Because we have no assumptions on the black-box renderer R', it can render photorealistic images at a high computational cost, by enabling interreflection, occlusion and rich illumination models *etc*. such that the final image is an accurate estimate under real-world physics as if captured by a real camera. In this case, the transferability of "adversarial meshes" generated by *meshAdv* is crucial since we want to avoid the expensive computation in R' and still be able to generate such S^{adv}.

We analyze two scenarios for such transferability.

Controlled Rendering Parameters Before black-box attacks, we want to first test our "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 model g as expected, we would like to see whether $I'^{\text{adv}} = R'(S^{\text{adv}}; P, L)$ can still fool the model g.

Unknown Rendering Parameters In this scenario, we would like to use *meshAdv* to attack a non-differentiable system $g(R'(S; P^*, L^*))$ under fixed, unknown rendering parameters P^*, L^*. In practice, we will have access to the mesh S and its mask M in the original photorealistic rendering $I' = R'(S; P^*, L^*)$, as well as the model g. Directly transfer from one renderer to another may not work due to complex rendering conditions. To improve the performance of such black-box attack, we propose a pipeline as follows:

1. Estimate camera parameters $\hat{P}$ by reducing the error of object silhouette $\|R_{\text{mask}}(S; P) - M\|^2$, where $R_{\text{mask}}(S; P)$ renders the mask of the object S (lighting is irrelevant to produce the mask);

2. Given $\hat{P}$, estimate lighting parameters $\hat{L}$ by reducing the masked error of rendered images: $\|M \circ (R(S; \hat{P}, L) - I')\|^2$, where the operator $\circ$ is Hadamard product;

3. Given $\hat{P}, \hat{L}$, use *meshAdv* to generate the "adversarial mesh" S^{adv} such that $R(S^{\text{adv}}; \hat{P}, \hat{L})$ fools g; To improve robustness, we add random perturbations to $\hat{P}$ and $\hat{L}$ when optimizing;

4. Test S^{adv} in the original scene with photorealistic renderer R': obtain the prediction $g(R'(S^{\text{adv}}; P^*, L^*))$.

6. Experimental Results

In this section, we first show the attack effectiveness of "adversarial meshes" generated by *meshAdv* against classifiers under different settings. We then visualize the perturbation flow of vertices to better understand the vulnerable regions of those 3D objects. User studies show that the proposed perturbation is subtle and will not mislead human recognition. In addition, we show examples of applying *meshAdv* to object detectors in physically realistic scenes. Finally, we evaluate the transferability of "adversarial meshes" generated by *meshAdv* and illustrate how to use such transferability to attack a black-box renderer.

6.1. Experimental Setup

For victim learning models g, we choose DenseNet [22] and Inception-v3 [48] trained on ImageNet [12] for classification, and Yolo-v3 trained on COCO [29] for object detection. For meshes (S), we preprocess CAD models in PASCAL3D+ [52] using uniform mesh resampling with MeshLab [9] to increase triangle density. Since these 3D objects have constant texture values, for texture perturbation we also start from constant as pristine texture.

For the differentiable renderer (R), we use the off-the-shelf PyTorch implementation [26, 39] of the Neural Mesh Renderer (NMR) [24] to generate "adversarial meshes". For rendering settings ($R(\cdot; P, L)$) when attacking classifiers, we randomly sample camera parameters P and lighting parameters L, and filter out configurations such that the classification models have 100% accuracy when rendering pristine meshes. These rendering configurations are then fixed for evaluation, and we call meshes rendered under these configurations *PASCAL3D+ renderings*. In total, we have

Perturbation Type	Model	Test Accuracy	Best Case		Average Case		Worst Case	
			Avg. Distance	Succ. Rate	Avg. Distance	Succ. Rate	Avg. Distance	Succ. Rate
Shape	DenseNet	100.0%	8.4×10^{-5}	100.0%	1.8×10^{-4}	100.0%	3.0×10^{-4}	100.0%
	Inception-v3	100.0%	4.8×10^{-5}	100.0%	1.2×10^{-4}	99.8%	2.3×10^{-4}	98.6%
Texture	DenseNet	100.0%	3.8×10^{-3}	100.0%	1.1×10^{-2}	99.8%	2.6×10^{-2}	98.6%
	Inception-v3	100.0%	3.7×10^{-3}	100.0%	1.3×10^{-2}	100.0%	3.2×10^{-2}	100.0%

Table 1: Attack success rate of *meshAdv* and average distance of generated perturbation for different models and different perturbation types. We choose rendering configurations in *PASCAL3D+ renderings* such that the models have 100% test accuracy on pristine meshes so as to confirm the adversarial effects. The average distance for shape based perturbation is computed using the 3D Laplacian loss from Equation 5. The average distance for texture based perturbation is the root-mean-squared error of face color change.

7 classes, and for each class we generate 72 different rendering configurations. More details are shown in the supplementary material.

For optimizing the objective, we use Adam [25] as our solver. In addition, we select the hyperparameter λ in Equation 1 using binary search, with 5 rounds of search and 1000 iterations for each round.

6.2. *MeshAdv* on Classification

In this section, we evaluate quantitative and qualitative performance of *meshAdv* against classifiers.

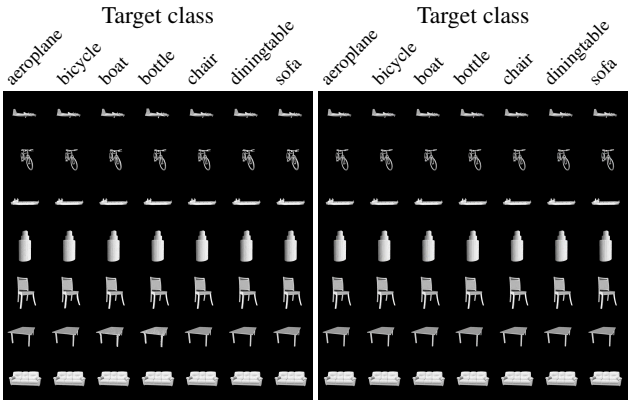
For each sample in our *PASCAL3D+ renderings*, we try to targeted-attack it into the other 6 categories. Next, for each perturbation type (shape and texture) and each model (DenseNet and Inception-v3), we split the results into three different cases similar to [5]: *Best Case* means we attack samples within one class to other classes and report on the target class that is *easiest* to attack. *Average Case* means we do the same but report the performance on *all* of the target classes. Similarly, *Worst case* means that we report on the target class that is *hardest* to attack. The corresponding results are shown in Table 1, including attack success rate of *meshAdv*, and the evaluation on generated shape and texture based perturbation respectively. For shape based perturbation, we use the Laplacian loss from Equation 5 as the distance metric. For texture based perturbation, we compute the root-mean-square distance of texture values for each face of the mesh: $\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{t}_i^{\text{adv}} - \boldsymbol{t}_i)^2}$, where $\boldsymbol{t}_i$ is the texture color of the i-th face among the mesh's total m faces. The results show that *meshAdv* can achieve almost 100% attack success rate for either adversarial perturbation types.

Figure 2 shows the generated "adversarial meshes" against Inception-v3 after manipulating the vertices and texture respectively. The diagonal shows the images rendered with the pristine meshes. The target class of each "adversarial mesh" is shown at the top, and similar results for DenseNet are included in the supplementary material. Note that the samples in the image are randomly selected and not manually curated. It is worth noting that the perturbation on object shape or texture, generated by *meshAdv*, is barely noticeable to human, while being able to mislead classifiers. To help assess the vertex displacement in shape perturbation, we discuss the flow visualization and human perceptual study in the following paragraphs.

Visualizing Vertex Manipulation In order to better understand the vulnerable regions of 3D objects, in Figure 3, we visualize the magnitude of the vertex manipulation flow using heatmaps. The heatmaps in the figure correspond to the ones in Figure 2(a). We adopt two viewpoints in this figure: the rendered view (i), which is the same as the one used for rendering the images; and the canonical view (ii), which is achieved by fixing camera parameters for all shapes: we set the azimuth angle to $135°$ and the elevation angle to $45°$. From the heatmaps we observe that the regions with large curvature value and close to the camera (such as edges) are more vulnerable, as shown in the example in Figure 3(d). We find 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 to change drastically.

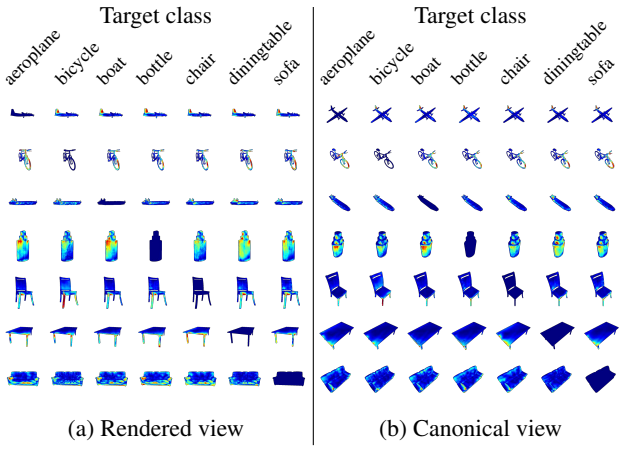
In addition to magnitude, we additionally show an example of flow directions in Figure 3(c), which is a close-up 3D quiver plot of the vertex flow in the vertical stabilizer region of an aeroplane. In this example, the perturbed aeroplane mesh is classified to "bicycle" in its rendering. From this figure, we observe that the adjacent vertices tend to flow towards similar directions, illustrating the effect of our 3D Laplacian loss operated on vertex flows (Equation 5).

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 which are misclassified by DenseNet and Inception-v3. Participants were asked to recognize those adversarial object to one of the two classes (the ground-truth class and the adversarial target class). The

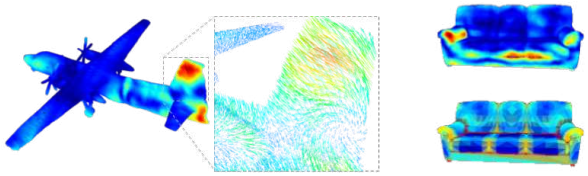


(a) Perturbation on shape (b) Perturbation on texture

Figure 2: Benign images (diagonal) and corresponding adversarial examples generated by *meshAdv* on *PASCAL3D+ renderings* tested on Inception-v3. Adversarial target classes are shown at the top. We show perturbation on (a) shape and (b) texture. Similar results for DenseNet can be found in the supplementary material.



(a) Rendered view (b) Canonical view



(c) Vertex flow of an "adversarial mesh" targeting "bicycle" (d) Perturbation (top) vs curvature (bottom)

Figure 3: (a) and (b) are visualization of shape based perturbation with respect to Figure 2(a). (c) is a close view of flow directions, and (d) is an example to compare the magnitude of perturbation with the magnitude of curvature. Warmer color indicates greater magnitude and vice versa.

order of these two classes was randomized and the adversarial objects are appeared for 2 seconds in the middle of the screen during each trial. After disappearing, the participant has unlimited time to select the more feasible class according to their perception. For each participant, one could only conduct at most 50 trials, and each adversarial image was shown to 5 different participants. The detailed settings of our human perceptual study are described in the supplementary material. In total, we collect 3820 annotations from 49 participants. In $99.29 \pm 1.96\%$ of trials the "adversarial meshes" were recognized correctly, indicating that our adversarial perturbation will not mislead human as they can almost always assign the correct label of these "3D adversarial meshes".

Multiview Robustness Analysis In addition to a fixed camera when applying *meshAdv*, we also explore the robustness of *meshAdv* against a range of viewpoints for shape based perturbation. First, we create a victim set of images rendered under 5, 10 or 15 different azimuth angles for optimizing the attack. We then sample another 20 unseen views within the range for test. The results are shown in Table 2. We can see that the larger the azimuth range is, the harder to achieve high attack success rate. In the meantime, *meshAdv* can achieve relatively high attack success rate when more victim instances are applied for training. As a result, it shows that the attack robustness can potentially be improved under various viewpoints by optimizing on large victim set.

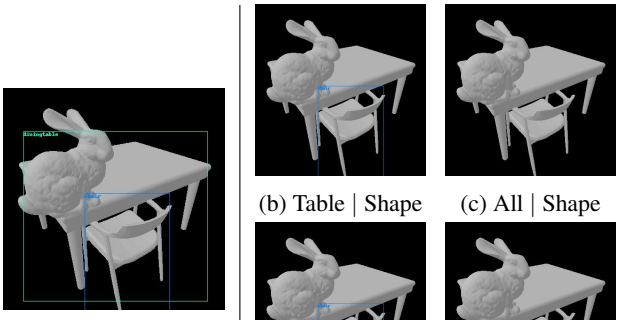
Victim Set Size	Azimuth Range		
	$45^\circ \sim 60^\circ$	$35^\circ \sim 70^\circ$	$15^\circ \sim 75^\circ$
5 views	67%	45%	28%
10 views	73%	58%	38%
15 views	79%	74%	48%

Table 2: Targeted attack success rate for unseen camera views. We attack using 5, 10, or 15 views, and test with 20 unseen views in the same range.

6.3. *MeshAdv* on Object Detection

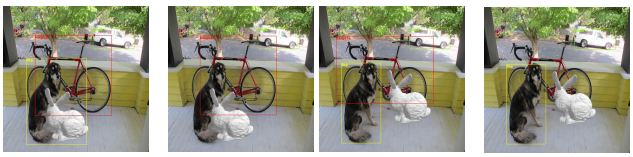
For object detection, we use Yolo-v3 [42] as our target model.

Indoor Scene First, we test *meshAdv* within the indoor scene which is pure synthetic. We compose the scene manually with a desk and a chair to simulate an indoor setting, and place in the scene a single directional light with low ambient light. We then put the Stanford Bunny mesh [49] onto the desk, and show that by manipulating either the shape or the texture of the mesh, we can achieve the goal of either removing the target table detection or removing all detections while keeping the perturbation almost unnoticeable,



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

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



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

Figure 5: "Adversarial meshes" generated by *meshAdv* for an outdoor photo. (a) and (c) show images rendered with pristine meshes as control experiments, 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 respectively.

as shown in Figure 4.

Outdoor Scene Given a real photo of an outdoor scene, we hope to remove the detections of real objects in the photo. Different from the indoor sceen in which lighting is known, we have to estimate the parameters of a sky lighting model [21] using the API provided by Hold-Geoffroy et al. [20] as groundtruth lighting and adapt to the differentiable renderer. We then use this lighting to render our mesh onto the photo. In the real photo, we select the dog and the bicycle as our target objects and aim to remove the detection one at a time. We show that we successfully achieve the adversarial goal with barely noticeable perturbation, as in Figure 5.

6.4. Transferability to Black-Box Renderers

As mentioned in Section 5, the final adversarial goal is to black-box attack a system $g(R'(S; P, L))$ in which the renderer R' is a computationally intensive renderer that is able to produce photorealistic images. Here we choose Mitsuba [23] as such renderer, and focus on shape based perturbation.

Model/Target	aeroplane	bicycle	boat	bottle
DenseNet	65.2%	69.1%	66.7%	63.0%
Inception-v3	67.1%	83.3%	39.6%	76.9%
Model/Target	**chair**	**diningtable**	**sofa**	**average**
DenseNet	37.1%	70.3%	47.9%	59.8%
Inception-v3	32.1%	75.0%	52.3%	60.9%

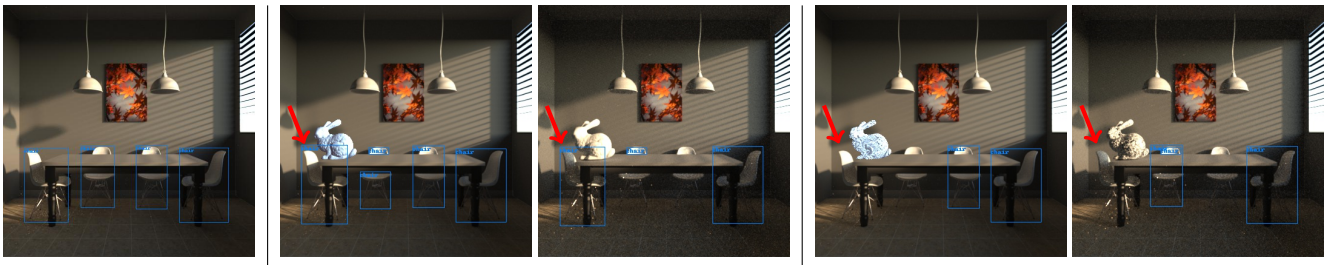
Table 3: Untargeted attack success rate against Mitsuba by transferring "adversarial meshes" generated by attacking a differentiable renderer targeting different classes.

Figure 6: Confusion matrices of targeted success rate for evaluating transferability of "adversarial meshes" on different classifiers. **Left**: DenseNet; **right**: Inception-v3.

Controlled Rendering Parameters Before perform such attacks, we first evaluate the transferability under controlled parameters. We directly render the "adversarial meshes" S^{adv} generated in Section 6.2 using Mitsuba, with the same lighting and camera parameters. We then calculate the targeted/untargeted attack success rate by feeding the Mitsuba-rendered images to the same victim classification models g. The result of untargeted attacks are shown in Table 3, and the confusion matrices for targeted attacks are show in Figure 6. We observe that for untargeted attack, the "adversarial meshes" can be transferred to Mitsuba with relatively high atttack success rate for untargeted attack; while as shown in Figure 6, the targeted attack barely transfers in this straightforward setting.

Unknown Rendering Parameters To more effectively targeted attack the system $g(R'(S; P^*, L^*))$ when rendering parameters P^*, L^* are unknown, we apply the pipeline from Section 5 on a classifier and an object detector, respectively. we first use the Adam optimizer [25] to obtain the camera estimate $\hat{P}$, then estimate the lighting L^* using 5 directional lights and an ambient light $\hat{L}$. Note that

Figure 7: Transferability of "adversarial meshes" against classifiers in unknown rendering environment. We estimate the camera viewpoint and lighting parameters using the differentiable renderer NMR, and apply the generated "adversarial mesh" to the photorealistic renderer Mitsuba. The "airliner" is misclassified to the target class "hammerhead" after rendered by Mitsuba.



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

Figure 8: Transferability of "adversarial meshes" against object detectors in unknown rendering environment. (b) (c) are controlled experiments. S^{adv} is generated using NMR (d), targeting to hide the leftmost chair (see red arrows), and the adversarial mesh is tested on Mitsuba (e). We use "S/S^{adv} | renderer" to denote whether the added object is adversarially optimized and the renderer that we aim to attack with transferability respectively.

the groundtruth lighting L^* spatially varies due to inter-reflection and occlusion, so it is impossible to have an exact estimate using the global lighting model in NMR. Then we manipulate the shape S^{adv} in the NMR until the image $I^{\text{adv}} = R(S^{\text{adv}} : \hat{P}, \hat{L})$ can successfully targeted-attack the classifier or the object detector g with a high confidence. During this process, we add small random perturbation to the estimated parameters $(\hat{P}, \hat{L})$ such that S^{adv} will be more robust under uncertainties. For testing, we re-render S^{adv} with Mitsuba using the original setting and test the rendered image $I'^{\text{adv}} = R'(S^{\text{adv}}, P^*, L^*)$ on the same model g.

For classification, we place an aeroplane object from PASCAL3D+ and put it in an outdoor scene under sky light. As is shown in Figure 7, we successfully attacked the classifier to output the target "hammerhead" by replacing the pristine mesh with our "adversarial mesh" in the original scene. Note that even we do not have an accurate lighting estimate, we still achieve the transferability by adding perturbation to lighting parameters. For object detection, we modified a scene from [3], and placed the Stanford Bunny object into the scene. The adversarial goal here is to remove the *leftmost* chair in the image. Without an accurate lighting estimate, Figure 8 shows that the "adversarial meshes" can still successfully remove the target (the leftmost chair) from the detector.

7. Conclusion

In this paper, we proposed *meshAdv* to generate "adversarial meshes" by manipulating the shape or the texture of a mesh. These "adversarial meshes" can be rendered to 2D domains to mislead different machine learning models. We evaluate *meshAdv* quantitatively and qualitatively using CAD models from PASCAL3D+, and also show that the adversarial behaviors of our "adversarial meshes" can transfer to black-box renderers. This provides us a better understanding of adversarial behaviors of 3D meshes in practice, and can motivate potential future defenses.

Acknowledgement We thank Lei Yang, Pin-Yu Chen for their valuable discussions on this work. This work is partially supported by the National Science Foundation under Grant CNS-1422211, CNS-1616575, IIS-1617767 and DARPA under Grant 00009970.

References

[1] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 284–293. JMLR.org, 2018. 1, 2

[2] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015. 2

[3] B. Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 8

[4] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017. 1

[5] N. Carlini and D. 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*, pages 39–57, 2017. doi: 10.1109/SP.2017.49. URL https://doi.org/10.1109/SP.2017.49. 1, 5

[6] F. Cayre and B. Macq. Data hiding on 3-d triangle meshes. *IEEE Transactions on Signal Processing*, 51(4):939–949, April 2003. ISSN 1053-587X. doi: 10.1109/TSP.2003.809380. 3

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. 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. 2

[8] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen. Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV)*, 2015. 2

[9] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136, 2008. 4

[10] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 1

[11] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005. 3

[12] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848. 4

[13] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. D. Williams, et al. Recent advances in deep learning for speech research at microsoft. In *ICASSP*, volume 26, page 64, 2013. 1

[14] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1, 2017. 1, 2

[15] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song. Physical adversarial examples for object detectors. *arXiv preprint arXiv:1807.07769*, 2018. 3

[16] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[18] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding realworld indoor scenes with synthetic data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4077–4085, 2016. 2

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[20] Y. Hold-Geoffroy, K. Sunkavalli, S. Hadap, E. Gambaretto, and J.-F. Lalonde. Deep outdoor illumination estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017. 7

[21] L. Hosek and A. 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. 7

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 1, 4

[23] W. Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 7

[24] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 4

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 7

[26] N. Kolotouros. Pytorch implememtation of the neural mesh renderer. https://github.com/daniilidis-group/neural_renderer, 2018. Accessed: 2018-09-10. 2, 4

[27] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1, 2

[28] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 2

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[30] H.-T. D. Liu, M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *International Conference on Learning Representations*, 2019. 2

[31] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *Computer Vision – ECCV 2014*, pages 154–169, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10584-0. 2

[32] D. Marr. *Vision: A Computational Investigation into the Hu-*

man Representation and Processing of Visual Information. Henry Holt and Co., Inc., New York, NY, USA, 1982. ISBN 0716715678. 3

[33] F. Massa, B. Russell, and M. Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[34] J. McCormac, A. Handa, S. Leutenegger, and A. 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. 2

[35] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. 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. 1, 2

[36] A. Mordvintsev, N. Pezzotti, L. Schubert, and C. Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations. 2

[37] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems*, pages 7891–7901. 2018. 2

[38] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016. 1, 2

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 4

[40] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 49–56, 1999. ISBN 0-201-48560-5. doi: 10.1145/311535.311540. 3

[41] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 497–500. ACM, 2001. ISBN 1-58113-374-X. doi: 10.1145/383259.383317. 2

[42] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2, 3, 6

[43] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016. 2

[44] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484, 2016. 1

[45] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[46] H. Su, C. R. Qi, Y. Li, and L. 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. 2

[47] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1, 4

[49] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. doi: 10.1145/192161.192241. 6

[50] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *CVPR*, 2017. 2

[51] C. R. Vogel and M. E. Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996. 3

[52] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 75–82. IEEE, 2014. 2, 4

[53] C. Xiao, R. Deng, B. Li, F. Yu, D. Song, et al. Characterizing adversarial examples based on spatial consistency information for semantic segmentation. In *Proceedings of the (ECCV)*, pages 217–234, 2018. 1

[54] C. Xiao, B. Li, J. yan Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/543. URL https://doi.org/10.24963/ijcai.2018/543. 2

[55] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018. 1, 2

[56] D. Yang and J. Deng. Shape from shading through shape evolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[57] A. K. F. Y. L. Z. X. T. J. X. Z. Wu, S. Song. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015. 2

[58] X. Zeng, C. Liu, W. Qiu, L. Xie, Y.-W. Tai, C. K. Tang, and A. L. Yuille. Adversarial attacks beyond the image space. *arXiv preprint arXiv:1711.07183*, 2017. 2, 3

[59] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2