DTA: Physical Camouflage Attacks using Differentiable Transformation Network

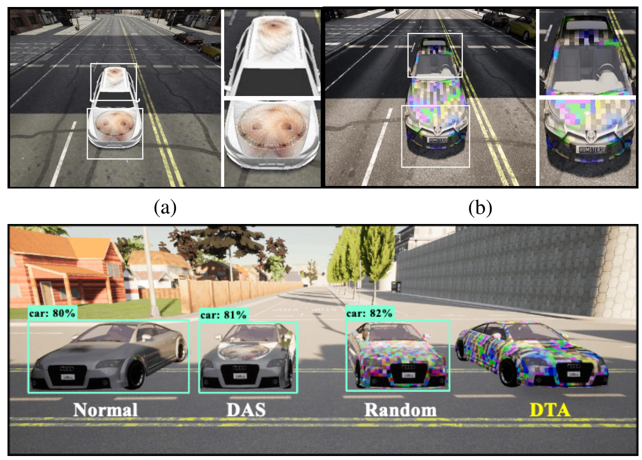
Naufal Suryanto [1,‡], Yongsu Kim [1,2,‡], Hyoeun Kang [1], Harashta Tatimma Larasati [1,4], Youngyeo Yun [1]
Thi-Thu-Huong Le [1,5], Hunmin Yang [3], Se-Yoon Oh [3], Howon Kim [1,2,*]
[1]Pusan National University, South Korea; [2]SmartM2M, South Korea; [3]Agency for Defense Development (ADD), South Korea;
[4]Institut Teknologi Bandung, Indonesia; [5]Hung Yen University of Technology and Education, Vietnam
https://islab-ai.github.io/dta-cvpr2022/

Abstract

To perform adversarial attacks in the physical world, many studies have proposed adversarial camouflage, a method to hide a target object by applying camouflage patterns on 3D object surfaces. For obtaining optimal physical adversarial camouflage, previous studies have utilized the so-called neural renderer, as it supports differentiability. However, existing neural renderers cannot fully represent various real-world transformations due to a lack of control of scene parameters compared to the legacy photo-realistic renderers. In this paper, we propose the Differentiable Transformation Attack (DTA), a framework for generating a robust physical adversarial pattern on a target object to camouflage it against object detection models with a wide range of transformations. It utilizes our novel Differentiable Transformation Network (DTN), which learns the expected transformation of a rendered object when the texture is changed while preserving the original properties of the target object. Using our attack framework, an adversary can gain both the advantages of the legacy photo-realistic renderers including various physical-world transformations and the benefit of white-box access by offering differentiability. Our experiments show that our camouflaged 3D vehicles can successfully evade state-of-the-art object detection models in the photo-realistic environment (i.e., CARLA on Unreal Engine). Furthermore, our demonstration on a scaled Tesla Model 3 proves the applicability and transferability of our method to the real world.

1. Introduction

Deep neural networks (DNNs), despite their renowned capability for solving computer vision tasks [10, 12, 19], have been proven vulnerable to adversarial examples [28].



(a) (b) (c)

Figure 1. **(a) Neural renderer used in prior work (Dual Attention Suppression (DAS)) [32].** Although it is differentiable, it possesses a limitation in physical properties representation (e.g., transparency for windshield) and lacks background blending (e.g., shadowing) because the object and the background scene are rendered separately. **(b) Our Differential Transformation Network (DTN).** DTN considers both differentiability and photo-realistic aspects by learning the correct transformation when the object's texture is changed. As shown, transparency for the windshield and shadowing at the bottom and the top of the car are rendered correctly. **(c) Comparison of detection results using different textures *(normal, DAS, random, DTA)* on photo-realistic environment.** Unlike other examples, the adversarial camouflage generated by our DTA framework successfully evades detection.

That is, carefully crafted inputs may cause DNN models to misrepresent a seemingly obvious image to the human eye, giving incorrect prediction results. A deliberate act by an adversary to take advantage of this weakness, namely the adversarial attack, has captured the attention of many in the past few years. Its potential applicability in not only the digital domain but also the physical domain has drawn significant interest.

‡ Equal contribution * Corresponding author

Compared to digital attacks, physical adversarial attacks are more difficult to launch since they must account for various physical constraints and conditions (e.g., lighting, camera pose, and occlusion). However, the fully physical attack experiments in the real world such as [2, 4, 15, 30] are extremely time consuming and expensive. Therefore, various studies have been conducted through simulation of the physical world in the digital environment by using legacy photo-realistic rendering software, such as Unreal Engine [8] and AirSim [26], facilitating parameter control. Examples of methods to craft physical adversarial camouflage, i.e., adversarial attack variant that focuses on hiding an object by fully covering the target object, in simulators can be found in [34, 35]. However, since such simulators are non-differentiable, the attacks employ a black-box approach such as utilizing a clone network [35] or a genetic algorithm [34], yielding an inevitably lower attack performance than the white-box counterpart.

To obtain the advantage of differentiability, more recent methods [7, 16, 32] have proposed the use of neural renderers for generating adversarial camouflage. However, the existing neural renderers (e.g., [17]) can only support the generation of foreground objects; hence, background images are still handled by the legacy photo-realistic renderers. As a result, they simply attach the generated target object to the background image, yielding inaccurate foreground-background blending effects, such as shadow casting and light reflection, as shown in Fig. 1a. Although workaround efforts, such as masking for handling occlusion [16], have been proposed, the overall adversarial camouflage results using existing neural renderers are still inferior in terms of photo-realistic attributes.

Motivated by the challenge faced in prior works, we develop an attack framework that takes advantage of the differentiability in neural renderers without compromising the photo-realistic properties of the target object. In particular, the framework leverages our novel neural rendering technique, which learns the representation of various scene properties (e.g., object material, lighting effects, and shadows) from legacy photo-realistic renderers. As a result, truly robust physical adversarial camouflage, verified from our experiment using both a photo-realistic simulation and a real-world example, can be obtained.

Our contributions can be summarized as follows:

- We present the Differentiable Transformation Attack (DTA), a framework for generating robust physical adversarial camouflage on 3D objects. It combines the advantages of a photo-realistic rendering engine with the differentiability of our novel rendering technique.
- We propose the Differentiable Transformation Network (DTN), a brand-new neural renderer that learns the transformations of an object when the texture is changed while preserving its original parameters for a realistic output that resembles the original.
- Our DTN can be embedded as an extension to provide differentiability to any rendering software (e.g., Unreal Engine [8]), enabling the use of any gradient-based method.
- We demonstrate that the adversarial camouflage generated from DTA is robust and applicable for evading pre-trained object detection models under various transformations in both simulations and the real world.
- Our attack method, DTA, outperforms previous works in terms of our evaluation of target object detection models and transferability to other models.

2. Related Works

Physical Adversarial Attack To launch an adversarial attack in the physical world, one of the most notable proposals is the Expectation Over Transformation (EOT) [2], which generates robust adversarial examples under various transformations, such as viewing distance, angle, and lighting condition. Most of the recent physical adversarial attack methods employ EOT-based algorithms to make the attack performance robust in the real world.

Adversarial Camouflage Physical perturbation [4] and patch-based methods [20, 22, 30] targeting planar and rigid objects were mainly proposed for real-world adversarial camouflage attacks. Subsequently, Huang et al. [15] proposed Universal Physical Camouflage (UPC) to generate adversarial camouflage that also covers non-planar and non-rigid objects. However, these methods can only be applied to a segment of an object and, due to their nature, can only attack at certain viewing angles.

A more recent approach of adversarial camouflage involves manipulating the color texture pattern of the target 3D object to degrade the detection performance of object detectors. This technique has the advantage of the ability of attack from any viewing angle by covering all parts of the object. Initially, black-box attack methods were commonly proposed since the rendering process, including texture mapping, is non-differentiable. For instance, Zhang et al. [35] proposed CAMOU, an adversarial camouflage method to hide vehicles from detectors by training a clone network that imitates both applying camouflage to vehicles and detecting the camouflaged vehicles. Meanwhile, Wu et al. [34] presented adversarial camouflage based on genetic algorithms to be applied on vehicle surfaces so that it is not recognizable by detectors in the CARLA simulator [6].

Neural Renderer-based Methods To gain white-box access —and in turn, improve the attack performance, there is a rising trend of leveraging the differentiability intrinsic in neural renderers for the adversarial camouflage generation, such as in [7, 16, 32]. For example, [32] proposed the

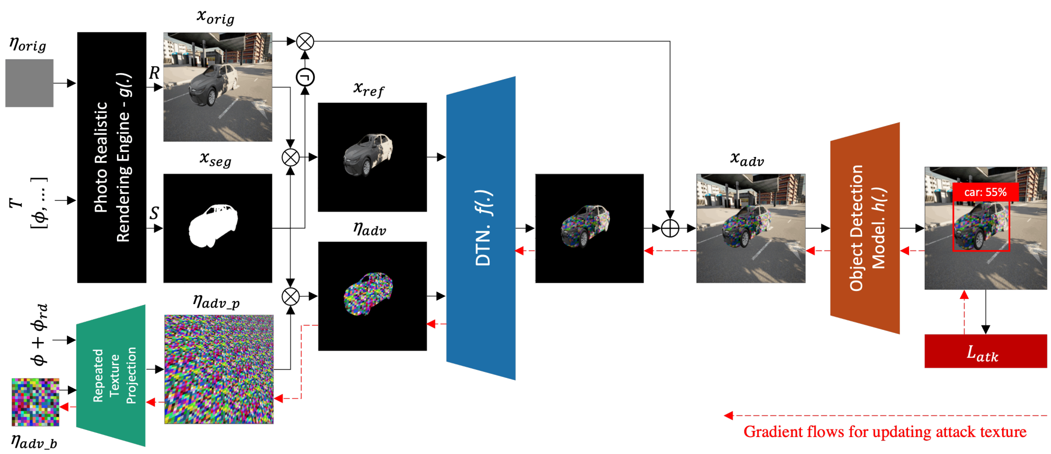


Figure 2. DTA framework for generating robust adversarial texture.

Dual Attention Suppression (DAS) attack to generate natural adversarial camouflage using a Neural 3D Mesh Renderer [17] by suppressing the model and human attention. However, existing neural renderers pose limitations on handling complex 3D interactions among scene properties. Inevitably, the resulting camouflage may not properly address the photo-realistic effects of the physical world.

To tackle these issues, we design an attack framework that combines the best of both worlds; that is, it has the differentiability of a neural renderer while retaining the photo-realistic attributes of the object. Our attack, namely the Differentiable Transformation Attack (DTA), utilizes our novel neural rendering model called Differentiable Transformation Network (DTN), which learns the representation of various scene properties from a legacy photo-realistic renderer, giving a more applicable solution in the physical world.

3. Methodology

In this section, we first describe DTN, our differentiable renderer that learns the expected transformation of projecting a specific pattern from a photo-realistic rendering engine. Furthermore, we describe DTA, the attack framework utilizing DTN for generating robust adversarial camouflage.

3.1. Problem Definition

The final goal of our proposed method is to generate a robust adversarial pattern to be applied to 3D objects in the simulated physical world to significantly degrade the detection score of that object in a variety of transformations, such as object materials, camera poses, lighting conditions, and background interactions.

Let h_θ be a hypothesis function for the object detection task, that satisfies $h_\theta(x) = y$. The notion x as the input denotes the 2D image (which includes the target object generated from the rendering process) and y as the output denotes the label of the detection result of the corresponding target object. The goal of our proposed method is to generate adversarial example x_{adv}, which satisfies $h_\theta(x_{adv}) \neq y$, by modifying the texture pattern of the target object. Suppose $L(h_\theta(x), y)$ is a loss function applied to h_θ that enables detection of an object in x as y. We can generate x_{adv} by solving Eq. 1.

$$\arg\max_{x_{adv}} L(h_\theta(x_{adv}), y) \tag{1}$$

Unlike 2D adversarial examples where we can directly modify the input image pixels, applying texture in 3D objects requires a rendering process with many parameters affecting the final image, such as shadows and light reflection.

Suppose R is a rendering function used in the photo-realistic rendering engine g, that satisfies Eq. 2,

$$R(T, \eta) = x \tag{2}$$

where T is a transformation matrix encoding various transformations, such as camera pose ϕ, lighting conditions, meshes, and material properties, as well as the target object and its location, whereas η is a texture that will be applied to the target object. If R is differentiable, we can find a robust adversarial texture η_{adv} that works in a wide variety of T using EOT [2] method. However, since R is not always differentiable, we propose a neural network f_ω that learns the texture transformations by solving Eq. 3,

$$f_\omega(x_{ref}, \eta_{exp}) = x_{ren} \tag{3}$$

where x_{ref} is the reference image obtained from the rendering function containing the transformation information,

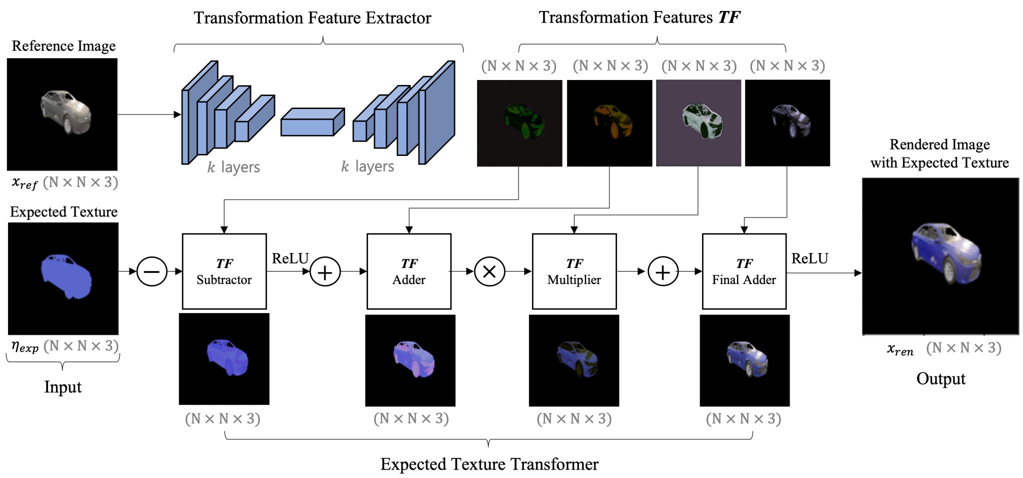


Figure 3. DTN architecture, network for learning expected transformation.

η_{exp} is the expected texture variable, and x_{ren} is the rendered image with the expected texture. When η_{exp} is the same texture used in x_{ref}, then $x_{ref} = x_{ren}$. Since f is differentiable, we can generate adversarial texture η_{adv}, which satisfies $f_\omega(x_{ref}, \eta_{adv}) = x_{adv}$, by solving Eq. 4.

$$\arg\max_{\eta_{adv}} L(h_\theta(f_\omega(x_{ref}, \eta_{adv})), y) \tag{4}$$

3.2. DTA Framework

The proposed DTA framework uses DTN, a neural network designed to solve the problem described in Eq. 3. DTN learns the transformation of the rendered object in x_{ren} given a reference image x_{ref} and expected texture η_{exp}. It relies on the photo-realistic image synthesized from a non-differentiable renderer to produce a differentiable version of the photo-realistic reference image after applying the expected texture. Furthermore, the differentiability of DTN is used to generate the robust adversarial pattern applied to the target object as the rendered adversarial texture to camouflage from the object detector. As a result, DTN provides a white-box ability for lowering specific attack loss. Depicted in Fig. 2, the DTA framework comprises four components: a photo-realistic rendering engine, a Repeated Texture Projection function, DTN, and the target object detection model.

Photo-Realistic Rendering Engine In our proposed DTA framework, the photo-realistic rendering engine is any software that can produce a photo-realistic image similar to the real physical world. One example is a game engine. It can be used for building a fully simulated physical world and synthesizing a photo-realistic image —even some can also synthesize semantic segmentation image, thanks to its detailed features and interactivity. In comparison, the interactivity has not been properly addressed in the existing differentiable renderers. Then, DTN is embedded as an extension to enable differentiability of the texture space, allowing adversarial texture generation for the target object.

DTN This technique uses photo-realistic RGB images synthesized from a rendering engine as the input reference image x_{ref}. The reference image only contains the masked target object where the expected texture η_{exp} will be applied. This masked image enables DTN to solely focus on learning and applying the transformation of the target object while ignoring the background.

In terms of architecture, DTN mainly consists of the Transformation Feature Extractor and Expected Texture Transformer, as illustrated in Fig. 3. The Transformation Feature Extractor is a convolutional autoencoder-like neural network that learns to extract transformation features of reference image x_{ref} and encode them as stacked transformation features TF. They are then sliced and used for transforming expected texture η_{exp} into a rendered image x_{ren}.

The final output of the Transformation Feature Extractor is TF. It has a $N \times N \times 12$ shape, where N is the resolution of input and output images and 12 is the last channel representing the four stacked RGB transformation features used to transform the expected texture to the image rendered by the Expected Texture Transformer. TF will have the same value no matter what the expected texture is. The idea is to prevent the Transformation Feature Extractor from overfitting because the expected texture is designed to never be shown directly as input to the network.

The transformation of η_{exp} into x_{ren} is performed using basic mathematical operations, such as subtraction, addition, and multiplication by each slice of TF. The design of this expected texture transformer is assumed to cover all

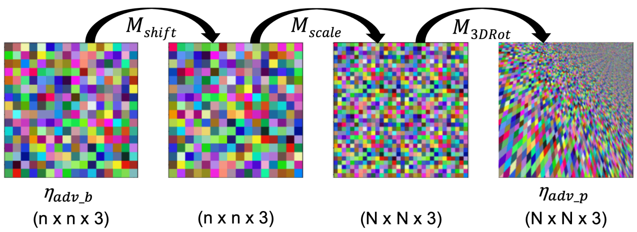


Figure 4. Sequence of transformation in Repeated Texture Projection Function.

kinds of basic transformations.

Repeated Texture Projection Function We propose a repeated pattern as our final attack camouflage texture. The repeated pattern has several advantages, such as ease of application because the texture can be used to cover the object while ignoring the texture mapping. The repeated pattern will also look more similar when we vary the viewing angle, creating a more robust attack.

In Figs. 2 and 4, we define an $n \times n$ adversarial pattern η_{adv_b} that will be transformed into η_{adv_p} until it has the same size as $N \times N$ DTN input requirement. We propose a Repeated Texture Projection function for simply projecting the repeated pattern based on the same camera pose ϕ used by the photo-realistic rendering engine. We add random pose ϕ_{rd} for adding a variety of transformations when it is used to generate the adversarial pattern, which is expected to increase the attack texture robustness and anticipate projection error. The Repeated Texture Projection function contains a sequence of operations for transforming the adversarial pattern with transformation matrix M.

The transformation matrix M used by repeated texture projection function covers shift, scale, and 3D rotation on 2D image operations. We can write it as Eq. 5,

$$\eta_{adv\_p} = M_{3DRot} \cdot M_{scale} \cdot M_{shift} \cdot \eta_{adv\_b} \tag{5}$$

where M_{shift} is a shift operation that determines the pattern order or initial location, M_{scale} is a scale operation that determines how large the texture will be when it is resized, and M_{3DRot} is a 3D rotation operation that determines how the 2D texture is rotated along the 3D axis. Each matrix M is calculated or calibrated based on a given camera pose $\phi + \phi_{rd}$ such that the projection covers the majority of the target's flat surface. Fig. 4 describes the result of each transformation sequence. For filling points outside boundaries, we use wrap mode, which extends the output by wrapping around the opposite edge, giving a repeated texture effect.

3.3. Framework Procedure

DTN Model Training Before using DTA for generating the adversarial pattern, DTN is trained with the dataset generated by the photo-realistic rendering engine. We set the dataset with two inputs, x_{ref} and η_{exp}, and one output, x_{ren}. First, we select a set of random flat color texture and predefined transformations. Then, we use the rendering engine to produce the photo-realistic images that will later be used as reference image x_{ref}, expected texture η_{exp}, and ground truth of rendered image x_{ren}. We use flat color texture as the expected texture so that there is no error caused by texture mapping during the training.

For each training example, x_{ref} only contains the masked target object applied with specific color and transformations. η_{exp} contains the flat color texture that will be applied to the target object and has a masked shape of the target object. We set the ground-truth of x_{ren} as the result of applying the flat color texture used in η_{exp} to the target object using the photo-realistic rendering engine. In this case, the same transformations used in x_{ref} except the texture are applied to the target object. Additionally, we utilize binary cross-entropy loss as per-pixel construction loss for training the DTN. The detailed algorithm of the DTN training process is described in the supplementary material.

DTA Attack Phase In the attack phase, the attack goal is to minimize the original target confidence score, which prevents the object detector from detecting the target object correctly. Since the object detection model outputs multiple boxes and class confidence scores, we just take the maximum target object confidence scores C and measure the log loss when we set the ground truth to zero. We can write the attack loss L_{atk} representing the above process as Eq. 6.

$$L_{atk}(h(x)) = \mathbb{E}_{t \sim T}[-log(1 - max(C(h(x))))] \tag{6}$$

$$\eta_{adv} = \arg\min_{\eta} L_{atk}(h(f_w(x_{ref}, \eta))) \tag{7}$$

Minimizing L_{atk} has the same effect as solving Eq. 4. We can use the differentiability of the full attack pipeline to find the best adversarial pattern η_{adv} that minimizes the attack loss by updating the η_{adv} based on the loss gradient. Eq. 7 describes the calculation of the best adversarial pattern η_{adv}. The full pipeline of the DTA framework for generating a robust attack pattern can be seen in Fig. 2 and Algorithm 1. In Algorithm 1, DTN f_w is the model that has completed the training process.

4. Experiments

4.1. Implementation Details

DTA Framework We utilize TensorFlow 2 [1] for implementing our DTA framework, except for the photo-realistic renderer, in which we use CARLA [6] simulator on Unreal Engine 4 [8]. CARLA provides ready-to-use APIs and digital assets (e.g., urban layouts, buildings, and vehicles) to simulate the physical world required for self-driving car research experiments. In our case, we tweak the original CARLA code to allow modification of car texture required

Algorithm 1: Generating attack texture using DTA

Input: Transformation set $T = \{t^{(1)}, \ldots, t^{(M)}\}$, Base flat color texture C, Rendering function R, Segmentation function S, Repeated texture projection function P, DTN f_w
Output: Attack texture η_{adv_b}
(1) Export X_{orig}, X_{seg}, and X_{ref} from the photo-realistic rendering engine
for $m = 1$ to M **do**
  $X_{orig}^{(m)} \leftarrow R(t^{(m)}, C)$
  $X_{seg}^{(m)} \leftarrow S(t^{(m)})$
  $X_{ref}^{(m)} \leftarrow X_{orig}^{(m)} \times X_{seg}^{(m)}$
end for
(2) Generate attack texture
Initialize η_{adv_b} with random values
for number of training iterations **do**
  Sample minibatch of each b samples $x_{ref} \in X_{ref}$, $x_{seg} \in X_{seg}$
  Derive ϕ corresponding to each x_{ref} from T
  $\eta_{adv\_p} \leftarrow P(\eta_{adv\_b}, \phi + \phi_{rd})$
  $\eta_{adv} \leftarrow \eta_{adv\_p} \times x_{seg}$
  Derive x_{adv} from $f_w(x_{ref}, \eta_{adv})$
  $x_{adv} = x_{adv} + (x_{orig} \times \neg x_{seg})$
  Calculate $L_{atk}(h(x_{adv}))$ by Eq. 6
  Update η_{adv_b} for minimizing $L_{atk}(h(x_{adv}))$ via backpropagation
end for

for dataset generation and texture evaluation. Also, we employ world-aligned texture in Unreal for repeated texture implementation instead of the original car's UV mapping.

Target Object We choose Toyota Camry as our target object for adversarial camouflage generation and evaluation, Audi TT for camouflage transferability evaluation, and Tesla Model 3 for real-world evaluation.

Datasets For DTN model training and evaluation, we select a map in CARLA and randomly choose 75 spawn locations for generating both training and validation datasets, and another 50 for generating testing datasets. We spawn the target car at each location and capture the images using cameras with 5m distance, 15-degree pitch, and every 45-degree rotation as a single transformation. For each transformation, we sequentially change the car texture with 50 random flat color textures. Our train and test datasets use different transformations and colors. For adversarial pattern generation, we select 250 spawn locations in the same map and generate datasets of the target object with the flat color texture used as the reference image during DTN training.

Evaluation Metrics To measure DTN's accuracy in predicting the transformation (with respect to the ground truth from the photo-realistic renderer), we use the model loss (i.e., binary cross-entropy) and mean squared error (MSE).

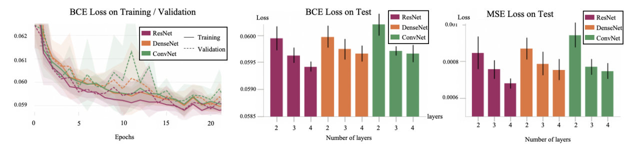


Figure 5. DTN model evaluations with different architectures [ResNet (red), DenseNet (orange), ConvNet (green)] and numbers of layers $k = [2, 3, 4]$.

Figure 6. Example of DTN prediction results.

To evaluate the performance of our proposed DTA, we use Average Precision@0.5, a commonly used metric for evaluating object detection models, including in the previous works. Furthermore, since our framework's main objective is to lower the target object's confidence score, it is also considered as another evaluation metric.

Target Models For ease of reproducibility, we choose the state-of-the-art COCO pre-trained object detection model [14] and employ both EfficientDetD0 [29] and YOLOv4 [3] as the target models. Furthermore, we evaluate the transferability of the generated pattern to other models (i.e., SSD [21], Faster R-CNN [25], and Mask R-CNN [9]).

Compared Methods We compare our adversarial camouflage with previous works on 3D physical attacks: CAMOU [35], ER [34], UPC [15], DAS [32], and an additional random pattern. However, UPC and DAS have different settings to recreate in our environment; thus, we only evaluate them on the transferability experiment.

DTN Parameters We employ a batch size of 32, 25 epochs, the Adam optimizer [18], and the same random seeds on each trial as the fixed parameters for training all networks. The input and output size of DTN are 512×512×3 to match the target object detection input size.

DTA Framework Parameters We select the 16×16 texture size as the adversarial camouflage following CAMOU's best implementation. We employ batch size of 32 for generating adversarial camouflage on EfficientDetD0 and 16 for YOLOv4. Each generation uses 200 epochs.

4.2. DTA Experiments

DTN Evaluation We employ $k = [2, 3, 4]$ layers for the Transformation Feature Extractor and utilize either residual [11] or dense connection [13] on the encoder besides a plain CNN. Fig. 5 shows that DTN with residual connection has the overall lowest average test loss and MSE, followed by DTN with dense connection. In our experiments, increasing the number of layers to four lowers the overall model loss,

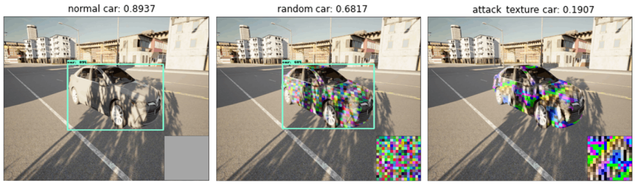


Figure 7. DTN texture rendering results with detection. From left to right: normal texture (detected as car), random texture (detected as car), attack texture (mis-detection). The car's confidence score for attack textures is the lowest at 0.1907.

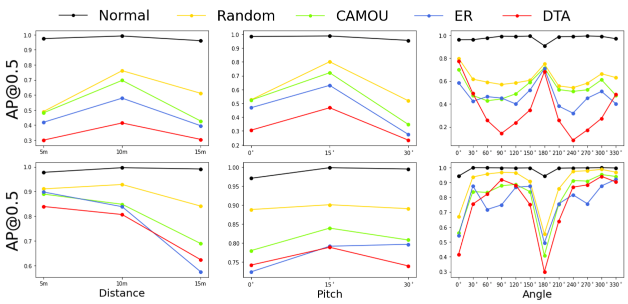


Figure 8. Average Precision@0.5 of various rendered textures with different camera poses on the photo-realistic simulator. First row: EfficientDetD0; Second row: YOLOv4.

with the best model achieving a 0.05942 average test model loss and 0.00068 average test MSE, resulting in an accurate rendering prediction as shown in Fig. 6.

Adversarial Camouflage Generation During the adversarial camouflage generation, we use DTN with ResNet and k of 4. We first generate a random pattern texture and optimize it to lower the attack loss using the full pipeline of the DTA framework. Fig. 7 shows how the texture is rendered and detected by DTA. As shown, after the completion of attack camouflage generation, the target confidence score is much lower, which may cause miss-detection. Moreover, it can be seen that a sole random pattern texture is not sufficient for camouflaging the object detector.

Comparison on Photo-Realistic Simulator We perform a comparative experiment to evaluate our adversarial camouflage on CARLA simulator. We generate evaluation datasets with 200 random locations from four different simulated towns with camera settings: distance (5 m, 10 m, 15 m), pitch angles (0°, 15°, 30°), and a 30 degree rotation interval. This also evaluates whether the generated adversarial camouflage is robust enough to generalize the attack performance on an untrained transformation distribution.

As shown in Tab. 1, our attack pattern (in bold) lowers both the target AP@0.5 and confidence score more than the random texture or the other methods, such as CAMOU and ER, which consider the system as a black box. The details of the average performance for each camera pose can be seen in Fig. 8, which implies that our adversarial camouflage has the overall lowest AP score on all camera poses. Additionally, Fig. 9 shows the example of our adversarial pattern evaluation with different transformations on the photo-realistic simulator.

Table 1. Camouflage comparison on photo-realistic simulator. (↓) denotes performance drop with respect to normal texture.

Model	Texture	AP@0.5 (↓)	Conf. (↓)
EffDetD0 [29]	Normal	0.98	0.75
	Random	0.62 (0.36)	0.39 (0.36)
	CAMOU [35]	0.53 (0.44)	0.34 (0.41)
	ER [34]	0.46 (0.51)	0.31 (0.47)
	DTA	**0.34 (0.63)**	**0.27 (0.48)**
YOLOv4 [3]	Normal	0.99	0.96
	Random	0.89 (0.10)	0.76 (0.20)
	CAMOU [35]	0.81 (0.18)	0.64 (0.32)
	ER [34]	0.77 (0.22)	0.60 (0.35)
	DTA	**0.76 (0.23)**	**0.57 (0.38)**

Table 2. Transferability comparison on photo-realistic simulator.

Texture	Average Precision @0.5 (↓)		
	SSD [21]	Faster R-CNN [25]	Mask R-CNN [9]
Normal	0.85	0.94	0.94
Random	0.52 (0.34)	0.69 (0.25)	0.74 (0.19)
UPC [15]	0.66 (0.20)	0.82 (0.11)	0.90 (0.03)
DAS [32]	0.79 (0.06)	0.89 (0.04)	0.97 (-0.03)
CAMOU [35]	0.27 (0.59)	0.55 (0.39)	0.65 (0.29)
ER [34]	0.27 (0.59)	0.56 (0.38)	0.64 (0.30)
DTA	**0.18 (0.67)**	**0.41 (0.53)**	**0.56 (0.37)**

We further perform a comparative experiment to evaluate the transferability of our adversarial camouflage to another car (i.e., Audi TT), other transformations, and target models that are not used in the attack phase. We use the same camera settings as those used in the previous experiment. Specifically, UPC and DAS target their original paper's model while the others target EfficientDetD0.

As presented in Tab. 2, our attack also outperforms other camouflage methods in the transferability setting. Notably, UPC and DAS, which are patch-based camouflage methods, show lower performance compared to repeated pattern camouflage, which covers the car's entire paintable surface.

Physical Camouflage in Real World We conduct a real-world experiment by fabricating a 1:10 scaled Tesla Model 3 cars using a 3D printer. Due to limited production time and resources, we only fabricate two scaled models: one for the camouflaged vehicle with our DTA texture targeting EfficientDetD0, and another for representing a standard vehicle for reference. To achieve a realistic experiment, we place the scaled vehicles in real-life locations indoors and outdoors. We randomly select ten places and capture car images for every 45-degree interval with a similar distance

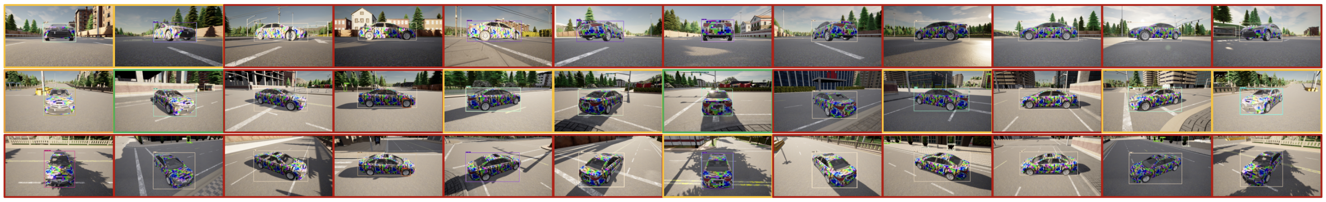


Figure 9. Adversarial camouflage evaluation on photo-realistic simulator. Camera angle is added by 30° for each column, and pitch by 15° for each row. Border: Red = misdetection; Yellow = partially correct (other labels also detected); Green = correct (detected as car).

Figure 10. Real-world evaluation using two 3D-printed scaled models of Tesla Model 3. The upper rows are the standard model while the bottom rows are the attack model.

and pitch using a Samsung Galaxy Note 20 Ultra. Fig. 10 illustrates how our camouflage can hide the car from the object detection model or result in misdetection while in contrast, the standard vehicle and the background objects are detected correctly. On evaluation, results on EfficientDetD0 and YOLOv4 for each vehicle model are presented on Tab. 3, which shows that our generated camouflage can successfully perform adversarial attack even in the real world.

5. Discussion

Implications DTN as a rendering method can easily be extended for use in texture transfer, e.g., [5, 24, 33], giving it potentially wide applicability, yet relatively harmless to the public. On the other hand, DTA as an attack certainly poses potentially dangerous consequences if it is to be implemented. Firstly, the attack pattern is more robust as it accounts for the photo-realistic effect, leading to an increase in attack success rate. This is concerning, particularly in the future era of autonomous vehicles. Secondly, the method of printing the adversarial pattern on vehicles rather than on traffic signs also adds to its hazard since painting cars can be done legally [35], whereas the latter is a violation and more likely to be removed by the authorities. On cars, it is easier for the adversary to launch the attack, and it can even be done in a coordinated fashion with other adversaries' cars. For mitigation, a possible solution is to reinforce object detection models with adversarial training [23, 31].

Table 3. Texture evaluation of vehicle model on real world.

Model	Texture	AP@0.5 (↓)	Conf. (↓)
EffDetD0 [29]	Normal	0.94	0.83
	DTA	**0.34 (0.60)**	**0.35 (0.48)**
YOLOv4 [3]	Normal	0.96	0.94
	DTA	**0.61 (0.35)**	**0.53 (0.41)**

Limitations DTN leverages a simple projection instead of a more sophisticated mapping for transferring the texture to the target object. Hence, an inaccurate texture may be produced for objects with a complex shape. For mitigation, we currently generate the pattern with random scaling, shifting, and rotation to improve robustness. Additionally, the current approach has not considered the naturalness of the pattern so that it does not look suspicious to a person, which may be important for some cases. This can be done by implementing additional loss, such as the smooth loss introduced in [27]. In the future, we plan to investigate further the internal texture mapping and naturalness of our adversarial camouflage approach.

6. Conclusion

In this paper, we proposed DTA, a framework that considers differentiability as well as photo-realistic aspects in its adversarial pattern generation, giving robust adversarial camouflage at any viewing angle. In particular, we leveraged our novel rendering technique, namely DTN, which can extract the expected transformation of a rendered object and retain its original attributes. Our experiments included a comparison with previous works (i.e., UPC [15], DAS [32], CAMOU [35], and ER [34]) in a photo-realistic simulator as well as a demonstration in the real world, showing the applicability and transferability of our approach.

Acknowledgments This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01343, Regional strategic industry convergence security core talent training business), and by a grant from Defense Acquisition Program Administration and Agency for Defense Development under contract UE201131RD.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018. 2, 3

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. 6, 7, 8

[4] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Robust physical adversarial attack on faster R-CNN object detector. *CoRR*, abs/1804.05810, 2018. 2

[5] Huong Quynh Dinh, Anthony Yezzi, and Greg Turk. Texture transfer during shape transformation. *ACM Trans. Graph.*, 24(2):289–310, apr 2005. 8

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2, 5

[7] Yexin Duan, Jialin Chen, Xingyu Zhou, Junhua Zou, Zhengyun He, Wu Zhang, and Zhisong Pan. Dpa: Learning robust physical adversarial camouflages for object detectors. *arXiv preprint arXiv:2109.00124*, 2021. 2

[8] Epic Games. Unreal engine, 2019. 2, 5

[9] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 6, 7

[10] 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*, pages 770–778, 2016. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[13] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 6

[14] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016. 6

[15] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L. Yuille, Changqing Zou, and Ning Liu. Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 720–729, June 2020. 2, 6, 7, 8

[16] Tingsong Jiang, Jialiang Sun, Weien Zhou, Xiaoya Zhang, Zhiqiang Gong, Wen Yao, Xiaoqian Chen, et al. Fca: Learning a 3d full-coverage vehicle camouflage for multi-view physical adversarial attack. *arXiv preprint arXiv:2109.07193*, 2021. 2

[17] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 2, 3

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[20] Mark Lee and J. Zico Kolter. On physical adversarial patches for object detection. *CoRR*, abs/1906.11897, 2019. 2

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 6, 7

[22] Xin Liu, Huanrui Yang, Linghao Song, Hai Li, and Yiran Chen. Dpatch: Attacking object detectors with adversarial patches. *CoRR*, abs/1806.02299, 2018. 2

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 8

[24] Yifang Men, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. A common framework for interactive texture transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 8

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 6, 7

[26] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 621–635, Cham, 2018. Springer International Publishing. 2

[27] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of*

the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery. 8

[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014. 1

[29] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020. 6, 7, 8

[30] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. *CoRR*, abs/1904.08653, 2019. 2

[31] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 8

[32] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8565–8574, 2021. 1, 2, 6, 7, 8

[33] Tuanfeng Y. Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph.*, 35(6):177:1–177:13, 2016. 8

[34] Tong Wu, Xuefei Ning, Wenshuo Li, Ranran Huang, Huazhong Yang, and Yu Wang. Physical adversarial attack on vehicle detector in the carla simulator. *CoRR*, abs/2007.16118, 2020. 2, 6, 7, 8

[35] Yang Zhang, Hassan Foroosh, Philip David, and Boqing Gong. Camou: Learning physical vehicle camouflages to adversarially attack detectors in the wild. In *International Conference on Learning Representations*, 2018. 2, 6, 7, 8