ACTIVE: Towards Highly Transferable 3D Physical Camouflage for Universal and Robust Vehicle Evasion

Naufal Suryanto1,‡, Yongsu Kim1,2,‡, Harashta Tatimma Larasati1, Hyoeun Kang1,2, Thi-Thu-Huong Le1, Yoonyoung Hong1, Hunmin Yang3, Se-Yoon Oh3, Howon Kim1,2,*

1Pusan National University, South Korea; 2SmartM2M, South Korea; 3Agency for Defense Development (ADD), South Korea

https://islab-ai.github.io/active-iccv2023/

Abstract

Adversarial camouflage has garnered attention for its ability to attack object detectors from any viewpoint by covering the entire object’s surface. However, universality and robustness in existing methods often fall short as the transferability aspect is often overlooked, thus restricting their application only to a specific target with limited performance. To address these challenges, we present Adversarial Camouflage for Transferable and Intensive Vehicle Evasion (ACTIVE), a state-of-the-art physical camouflage attack framework designed to generate universal and robust adversarial camouflage capable of concealing any 3D vehicle from detectors. Our framework incorporates innovative techniques to enhance universality and robustness, including a refined texture rendering that enables common texture application to different vehicles without being constrained to a specific texture map, a novel stealth loss that renders the vehicle undetectable, and a smooth and camouflage loss to enhance the naturalness of the adversarial camouflage. Our extensive experiments on 15 different models show that ACTIVE consistently outperforms existing works on various public detectors, including the latest YOLOv7. Notably, our universality evaluations reveal promising transferability to other vehicle classes, tasks (segmentation models), and the real world, not just other vehicles.

1. Introduction

Deep neural networks (DNNs) have achieved tremendous outcomes in a wide range of research fields, especially in computer vision, such as facial recognition and self-driving cars [8, 34, 7]. Despite their remarkable performance, DNNs, including object detection models, are vulnerable to adversarial attacks [2]. Generally, adversarial attacks can be classified into digital attacks and physical attacks [36]. Digital attacks are primarily carried out by adding small perturbations to pixels of input images. However, digital attacks have limitations in real-world scenarios because they have to manipulate the digital systems that may be configured with security schemes. To alleviate these limitations, physical attacks have been proposed that modify the object in the physical space rather than in the digital.

Nonetheless, physical attacks are more challenging due to the inherently complex physical constraints (camera pose, lighting, occlusion, etc.). Therefore, most physical attack methods exclude the perturbation constraint (i.e., the result can be suspicious). There are mainly two types of methods in physical attacks: adversarial patch and adversarial camouflage. The former [6, 32] is the method of physical...
attack by attaching a small, localized patch to an object. It only covers the planar part of the object’s surface and can fail to attack the detector depending on the viewing angles.

Adversarial camouflage methods [45, 41, 39, 35, 30] have been proposed to overcome the limitations of the adversarial patch. This approach covers the whole surface of the object by manipulating the texture of the object, which leads to better attack performance regardless of the viewing angles. Most of these methods use vehicles as the target objects due to their crucial role in real-world applications, such as surveillance systems and autonomous driving.

Due to the non-planarity of 3D vehicles, it is more challenging to generate optimal adversarial camouflage than in its 2D counterpart. Since the general 3D rendering is non-differentiable, early research [45, 41] employed a black-box approach to generating adversarial camouflage, inevitably yielding a lower attack performance than the white-box one. More recent research (e.g., DAS [39], FCA [35], and DTA [30]) utilize a neural renderer to acquire the advantage of the white-box approach, which offers differentiability. In particular, DTA proposes its own neural renderer capable of expressing various physical and realistic characteristics. Additionally, it employs its so-called Repeated Texture Projection function to apply the same attack texture to other vehicle types, thus improving universality. However, DTA relies on a simple texture projection, which may result in an inaccurate texture for non-planar shapes. For improving the accuracy and robustness of adversarial camouflage methods, it would be essential to bring forth a more sophisticated texture mapping approach.

In this paper, we propose **Adversarial Camouflage for Transferable and Intensive Vehicle Evasion (ACTIVE)**, a state-of-the-art adversarial camouflage framework that greatly enhances robustness, universality, and naturalness compared to previous methods, as shown in Fig. 1. Our contributions can be summarized as follows:

- **We utilize Triplanar Mapping**, a sophisticated texture mapping approach available through the neural renderer, to generate adversarial textures with improved robustness and universality. To the best of our knowledge, the use of this method in generating adversarial camouflage has never been found in literature.

- **We employ Stealth Loss**, our novel attack loss function that minimizes the detection score from all valid classes, resulting in the target vehicle being not only misclassified but also undetectable.

- **We improve the naturalness of the adversarial camouflage by utilizing larger texture resolutions than previous works** [45, 41, 30] and applying a smooth loss. Furthermore, we introduce a **Camouflage Loss** that can enhance the camouflage of the vehicle against the background.

- **Our extensive experiments demonstrate that ACTIVE consistently outperforms previous works, exhibiting improved universality from multiple perspectives**: instance-agnostic (available on various vehicle types), class-agnostic (even across different classes such as truck and bus), model-agnostic (performing on various vehicle detectors), task-agnostic (performing on segmentation models), and domain-agnostic (performing in real-world scenarios).

2. Related Works

The Expectation over Transformation (EoT) [4] has emerged as a leading approach in generating robust adversarial examples under various transformations, including variations in viewing distance, angle, and lighting conditions. Consequently, many adversarial camouflage methods [45, 41, 39, 35, 30] incorporate EoT-based algorithms to enhance their attack robustness in the physical scenarios.

Regarding the texture rendering process, differentiability is crucial to enable white-box attacks to obtain optimal adversarial camouflage, whereas non-differentiability of general texture rendering led to the initial proposal of the black-box approach. Zhang et al. [45] proposed CAMOU, utilizing the clone network that imitates the texture rendering and detection process, while Wu et al. [41] suggested finding optimal adversarial texture based on genetic algorithm.

Huang et al. [16] proposed the Universal Physical Camouflage Attack (UPC) as an alternative method for crafting universal adversarial camouflage, which differs from existing black-box approaches. To make UPC effective for non-rigid or non-planar objects, they introduced a set of transformations that can mimic deformable properties. However, subsequent studies [39, 35, 30] have found that the transferability of UPC is limited when it comes to various viewing angles, other models, and different environments due to its inherent limitations of the patch-based method.

More recent research used a neural renderer, which provides a differentiable texture rendering, to improve attack performance. Wang et al. [39] proposed the Dual Attention Suppression (DAS) attack, which suppresses both model and human attention. Meanwhile, Wang et al. [35] proposed the Full-coverage Camouflage Attack (FCA), which is more robust under complex views.

However, Suryanto et al. [30] pointed out that the DAS and FCA used a legacy renderer, which could not reflect various real-world characteristics and complex scenes, such as shadows and light reflections. They proposed the Differentiable Transformation Attack (DTA) that uses their own neural renderer, which provides rendering similar to a photo-realistic renderer. Despite its success, DTA also comes with several limitations. Notably, the method relies on a simple texture projection, which can lead to inaccurate texture mapping. Furthermore, the generated camouflage
features a colorful mosaic pattern, which appears unnatural to human observers. Tab. 1 compares how well existing approaches stack up against the suggested method under various criteria. As shown, our proposal satisfies and achieves all good values under each condition setting compared to prior methods.

Table 1: Comparison of proposed and existing physical camouflage attack methods.

<table>
<thead>
<tr>
<th>Attack</th>
<th>3D</th>
<th>WB</th>
<th>FC</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP [8]</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>**</td>
<td>*</td>
<td>-</td>
<td>**</td>
<td>-</td>
</tr>
<tr>
<td>UPC [16]</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>**</td>
<td>*</td>
<td>-</td>
<td>**</td>
<td>-</td>
</tr>
<tr>
<td>CM [45]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>*</td>
<td>-</td>
<td>η</td>
<td>-</td>
</tr>
<tr>
<td>ER [41]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
<td>-</td>
<td>**</td>
<td>-</td>
</tr>
<tr>
<td>DAS [39]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>-</td>
<td>*</td>
<td>**</td>
<td>-</td>
</tr>
<tr>
<td>FCA [35]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>η</td>
<td>-</td>
<td>η</td>
</tr>
<tr>
<td>DTA [30]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>η</td>
<td>-</td>
<td>η</td>
<td>**</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>η</td>
<td>η</td>
<td>η</td>
<td>η</td>
</tr>
</tbody>
</table>

Notes:
3D | White Box (WB) | Full Covering (FC)
(1) Universality (U): Might be universal, but only optimized on a single instance | Universal, the adversarial is optimized on multiple instances on the same category
(2) Applicability (A): Require exact position to place the adversarial | Can be placed anywhere which satisfies on the target object
(3) Naturalness (N): Consider naturalness such as using smooth texture |Require a more constrained setting
(4) Digital Transformation (D): Affine transformation only | Both affine transformation and brightness, contrast
(5) Physical Transformation (P): Camera position only | Both camera position and physical phenomena

3. Methodology

3.1. Problem Definition

Assume $f$ is a neural renderer that generates optimal adversarial camouflage through white-box attacks based on differentiable rendering. $f$ learns texture rendering by solving Eq. 1,

$$f(x_{ref}, \eta) = x_{ren}$$  \hspace{1cm} (1)

where $x_{ref}$ refers to the reference image, which includes the target vehicle, $\eta$ is the texture variable, and $x_{ren}$ is the rendered image with $\eta$. Previously, existing works have utilized neural renderer to generate adversarial camouflage by minimizing the loss function as shown in Eq. 2.

$$\arg \min_{\eta_{adv}} L(h(f(x_{ref}, \eta_{adv})), y)$$  \hspace{1cm} (2)

where $h$ is the hypothesis function for the vehicle detector, $y$ is the corresponding detection label output, and $L(h(x), y)$ is the loss function that represents the confidence score of $h$ regarding class $y$. Thus, solving Eq. 2 involves generating $\eta_{adv}$ to attack $h$ by minimizing its confidence score.

We have noticed that there are rooms for improvement in at least two aspects. First, most of the neural renderers employed in existing adversarial camouflage methods use object-dependent texture mapping, such as UV mapping. Therefore, if the vehicle type changes, the previously generated $\eta_{adv}$ cannot be used. Second, most existing methods only minimize the target class confidence score for $y$. While this may cause misclassification as a different class than $y$, the object detection itself may remain. Meanwhile, our proposal employs a method that applies an advanced and object-independent texture mapping approach (i.e., triplanar mapping), which later proves to solve the universality issues.

Furthermore, to address both limitations, we introduce Eq. 3,

$$\arg \min_{\eta_{adv}} \mathbb{E}_{v \in V, y \in Y} L(h(f(x_v, \eta_{adv})), y)$$  \hspace{1cm} (3)

where $V$ denotes the available vehicle types while $Y$ signifies the available classes of the detector. Solving Eq. 3 aims to: (1) improve universality by generating an attack pattern that is applicable to various vehicles simultaneously, and (2) enhance robustness by minimizing confidence scores for all valid classes to avoid detection as objects themselves.

3.2. ACTIVE Framework

To generate universal and robust adversarial camouflage, we propose the ACTIVE framework, which employs an object-independent texture mapping with a neural renderer and a new attack loss function to cause the vehicle undetectable. The overall framework is as illustrated in Fig. 2.

Triplanar Mapping (TPM). We introduce the use of triplanar mapping [24], a texture mapping technique that applies textures to objects by projecting them from three directions using their surface coordinate and surface normal that can be extracted from the depth image. We find this method particularly beneficial for generating object-independent adversarial textures since it does not rely on object-specific texture maps. Thus, we can optimize a common adversarial camouflage for multiple vehicles, making our attack instance-agnostic. Further, we introduce the Neural Texture Renderer (NTR), our improvement of DTN method by Suryanto et al. [30], which works well with our triplanar mapping while effectively preserving various physical characteristics. To our knowledge, we are the first to refine triplanar-mapped texture as the input to the neural renderer for enabling adversarial texture optimization across 3D instances. Our NTR enhances the efficiency of DTN by removing unnecessary elements of DTN, which the detail can be found in the Supplementary Material.

Stealth Loss. We propose stealth loss, a novel attack loss function for improving robustness. It considers two representative scores used in object detection models: the class confidence score and the objectness score. First, we minimize the objectness score, which determines the presence of an object in the detector, such as in YOLO families. Moreover, instead of minimizing the maximum confidence.
score for a specific class, we minimize the maximum confidence score across all classes, making our attack class-agnostic. This approach does not only mislead the model to misclassification, but also considers the possibility of the box being empty of objects. The attack loss $L_{atk}$ is written as Eq. 4,

$$h_d(x) = \begin{cases} h_c(x) \times h_o(x), & \text{if } \text{IoU}(h_b(x), gt) > t \\ 0, & \text{otherwise} \end{cases}$$

$$L_{atk}(x) = f_{log}(\max(h_d(x)))$$

where $x$ is an input image of a vehicle detector $h$, $h_c(x)$ is the confidence score of $h$ for $x$, $h_o(x)$ is the objectness score of $h$ for $x$, $h_b(x)$ is the detection result of $h$ for $x$ in the form of the bounding box, $\text{IoU}(h_b(x), gt)$ is the Intersection over Union (IoU) between the bounding box $h_b(x)$ and the ground-truth, $gt$, and $t$ is a custom IoU threshold. We define $h_d(x)$ as a detection score, which is the product of the confidence score and the objectness score. $f_{log}(n) = -\log(1 - n)$ is a log loss when the ground truth is zero. Then, minimizing $L_{atk}$ has the effect of minimizing both the confidence and objectness scores. Note that we only assign a value to the detection score for valid boxes with IoU greater than $t$, otherwise set it as zero. Thereby, the valid detection performance of the object detection model can be effectively lowered because the loss is applied only to the box that detects the object closely, and otherwise is excluded.

**Random Output Augmentation (ROA).** We propose an ROA module to enhance the texture robustness by various digital transformations [4]. Specifically, the module takes the output of the NTR, i.e., the adversarial example, and further augments it by applying random transformations such as scaling, translation, brightness, and contrast. While the NTR already provides robustness against various physical transformations, ROA allows for attaining an additional level of robustness with digital transformations to simulate changes happening in the real world to a certain extent, which to the best of our knowledge, have not been considered in most adversarial camouflage methods.

**Smooth Loss.** We utilize a smooth loss (i.e., Total Variation (TV) loss [22]) to improve the smoothness of the generated camouflage, which we define as $L_{sm}$ in Eq. 5,

$$L_{sm}(\eta) = \frac{1}{N_{sm}} \sum_{i,j} f_{log}(|\eta_{i,j} - \eta_{i+1,j}|) + f_{log}(|\eta_{i,j} - \eta_{i,j+1}|)$$

where $\eta_{i,j}$ is a pixel in a texture, $\eta$, at coordinate $(i, j)$ and $N_{sm} = (H-1) \cdot (W-1)$ is a scale factor with texture image height, $H$, and texture image width, $W$. We slightly modify the loss for scale adjustment and normalization compared to the original TV loss. That is, $L_{sm}$ is low when the values of adjacent pixels are close to each other. Thus, minimizing $L_{sm}$ improve the smoothness of the adversarial camouflage.

**Camouflage Loss.** We propose a camouflage loss to improve camouflage for human vision as well as computer vision. There are several ways to computationally measure camouflage effectiveness, the most common method being to measure the target-background similarity [33]. We use a method to extract the most dominant background colors and force the object texture color to be similar. First, we employ a camouflage function based on K-means clustering to ex-
tract the most dominant colors from all background images [43, 17, 42]. Next, we utilize Non-Printability Score (NPS) loss [28] to regulate the object texture color set. While it was originally proposed to craft a color set that is comprised mostly of colors reproducible by the printer, we replace the printable color set used in the original NPS loss with the most dominant background color set. The camouflage loss, $L_{cm}$, can be expressed as Eq. 6,

$$L_{cm}(\eta, B) = \frac{1}{N_{cm}} \sum_{i,j} f_{\text{log}}(\min_{b \in B} |b - \eta_{i,j}|)$$  \hspace{1cm} (6)

where $N_{cm} = H \cdot W$ is a scale factor of camouflage loss, and $B$ is the most dominant background color set. We can acquire the adversarial camouflage, which has a similar color to the background, by minimizing $L_{cm}$. Finally, our total loss, $L_{total}$, is constructed as Eq. 7,

$$L_{total} = \alpha L_{\text{atk}} + \beta L_{\text{sm}} + \gamma L_{cm}$$  \hspace{1cm} (7)

where $\alpha$, $\beta$, and $\gamma$ are the weights to control the contribution of each loss function. The full pipeline of the ACTIVE framework for generating adversarial camouflage by minimizing $L_{total}$ is illustrated in Fig. 2 and Alg. 1.

**Algorithm 1** ACTIVE adversarial camouflage generation

**Input:** Physical transformation set $T_p$, Digital transformation set $T_D$, Rendering function $R$, Segmentation function $S$, Depth function $D$, Triplanar mapping $P$, NTR $f$, ROA module $A$.

**Output:** Adversarial camouflage $\eta$

1. Export $x_{\text{ref}}, x_m, x_d, x_{\text{ref},m},$ and $x_{bg}$ from the rendering engine
   
   $x_{\text{ref}} \leftarrow R(T_p)$, $x_m \leftarrow S(T_p)$, $x_d \leftarrow D(T_p)$
   
   $x_{\text{ref},m} \leftarrow x_{\text{ref}} \times x_m$, $x_{bg} \leftarrow x_{\text{ref}} \times \neg x_m$

2. Export $B$ by Camouflage function $C$
   
   $B \leftarrow C(x_{bg})$

3. Generate adversarial camouflage $\eta$
   
   Initialize $\eta$ with random values

   for number of training iterations do
   
   Select the minibatch sample from $x_{\text{ref},m}, x_m, x_{bg}$
   
   Derive $\phi, \phi_d$ corresponding to each $x_{\text{ref},m}$ from $T_p$
   
   $\eta_p \leftarrow P(\eta, x_d, \phi + \phi_d)$
   
   $x_{adv,m} \leftarrow f(x_{\text{ref},m}, \eta_p)$
   
   $x_{adv} \leftarrow x_{adv,m} + x_{bg}$
   
   $x_{adv,a} \leftarrow A(x_{adv}, T_D)$
   
   Calculate $L_{\text{atk}}(x_{adv,a}), L_{\text{sm}}(\eta), L_{cm}(\eta, B)$ by Eq. 4, 5, 6
   
   Set $L_{total}$ by Eq. 7
   
   Update $\eta$ for minimizing $L_{total}$ via backpropagation

end for

4. Experiments

We perform comprehensive experiments to investigate the performance of our proposed method on multiple aspects, including robustness and universality. Each experiment and its comparison with previous works are designed with specific criteria. However, due to limited space, we only present the essential information and leave the detail to the Supplementary Materials.

4.1. Implementation Details

**Environment and Datasets.** We implement our attacking framework using TensorFlow 2 [1] and utilize CARLA [11] on Unreal Engine 4 (UE4) [12] as a physically-based simulator for training and evaluation, following [30]. We synthesize our own dataset for training and evaluation with various photo-realistic settings. We select five types of cars on CARLA for attack texture generation and robustness evaluation, and another different five for universality evaluation, where they are excluded from texture optimization. For NTR model training, a total of 50,625 and 150,000 photo-realistic images are used for training and testing, respectively. Regarding attack pattern generation, another 15,000 images for reference are employed. As for attack evaluation, we render the generated attack pattern on multiple cars using world-aligned texture in Unreal Engine to produce a repeated pattern while ignoring the texture UV Map, with 14,400 images for a single texture evaluation.

**NTR.** We build an NTR with four encoder-decoder layers using DenseNet architecture [15] (following [30]) and train using 20 epochs. Our experiment verifies that the network trained with only nine selected colors are able to generalize 50 random colors on the test, achieving 0.985 SSIM (comparable to 0.986 SSIM in the original DTN setting), but with 82% less data. Details in Supplementary Material.

**Attack Parameters.** Our attack texture is optimized using Adam [19] with 30 epochs. For ROA, we use 0.25 random brightness, $[0.75, 1.5]$ random contrast, and $[0.25, 1.0]$ random scale. For projection augmentation on triplanar mapping, we use $[-0.5, 0.5]$ random shift and $[-0.25, 0.25]$ random scale. For loss hyperparameters, we use $\alpha = 1.0$ with IoU threshold $t = 0.5$, $\beta = 0.25$, and $\gamma = 0.25$ as default. Also, we set the base texture resolution to $64 \times 64$.

4.2. Robustness Evaluations

**From Digital to Physical Simulation.** First, we perform a comparative experiment to investigate the effectiveness of existing rendering methods, including ours, by evaluating the performance of attack textures in the original pipeline compared to physical simulation (UE4). For a fair comparison with prior works, we follow DAS [39] and FCA [35] by selecting the same simulated town and Audi E-Tron car in CARLA, then optimizing the textures targeting YOLOv3.
Table 2: Digital-to-physical simulation comparison. Values are Average Precision@0.5 (%) of car in YOLOv3.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>UE4</td>
<td>-</td>
<td>99.57</td>
</tr>
<tr>
<td>DAS [39]</td>
<td>+ NMR</td>
<td>88.55</td>
<td>96.09</td>
</tr>
<tr>
<td>FCA [35]</td>
<td>+ NMR</td>
<td>52.05</td>
<td>92.28</td>
</tr>
<tr>
<td>DTA [30]</td>
<td>+ RTP &amp; DTN</td>
<td>16.91</td>
<td>41.95</td>
</tr>
<tr>
<td>Ours</td>
<td>+ TPM &amp; NTR</td>
<td>1.28</td>
<td>7.29</td>
</tr>
</tbody>
</table>

Figure 3: Rendering comparison of white-box 3D adv. camouflage methods: Adv. example from original pipeline vs. Fully transferred to physical simulation (UE4). Zoom in.

[25]. In detail, DAS and FCA utilize Neural Mesh Renderer (NMR) [18] to optimize the 3D car texture and attach it to the simulated town as background, while DTA [30] utilizes Repeated Texture Projection (RTP) and DTN to simply project repeated texture and render it to the reference image. Meanwhile, ours utilizes TPM and NTR for more accurate repeated texture mapping and rendering.

As shown in Tab. 2 and Fig. 3, adversarial example produced by our optimization pipeline results in a very high attack performance compared to related works, even after fully transferred to physical simulation. DAS exhibits poor performance due to its partial texture coverage which is aligned with [35, 30]. FCA can successfully evade the detection in the original pipeline but is fully detected in the physical simulation with a high score. Meanwhile, ours utilizes TPM and NTR for more accurate repeated texture mapping and rendering.

As shown in Tab. 2 and Fig. 3, adversarial example produced by our optimization pipeline results in a very high attack performance compared to related works, even after fully transferred to physical simulation. DAS exhibits poor performance due to its partial texture coverage which is aligned with [35, 30]. FCA can successfully evade the detection in the original pipeline but is fully detected in the physical simulation with a high score. Meanwhile, ours utilizes TPM and NTR for more accurate repeated texture mapping and rendering.

Table 3: Attack comparison on physically-based simulation. Values are Average Precision@0.5 (%) of car.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Single-Stage Detector</th>
<th>Two-Stage Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv3</td>
<td>SSD</td>
<td>FrRCNN</td>
</tr>
<tr>
<td>Normal</td>
<td>90.67</td>
<td>92.20</td>
</tr>
<tr>
<td>Random</td>
<td>70.01</td>
<td>79.01</td>
</tr>
<tr>
<td>Naive Cam.</td>
<td>60.26</td>
<td>59.48</td>
</tr>
<tr>
<td>DAS [39]</td>
<td>88.53</td>
<td>84.28</td>
</tr>
<tr>
<td>FCA [35]</td>
<td>76.92</td>
<td>76.35</td>
</tr>
<tr>
<td>CAMOU [45]</td>
<td>59.20</td>
<td>68.02</td>
</tr>
<tr>
<td>ER [41]</td>
<td>58.02</td>
<td>70.77</td>
</tr>
<tr>
<td>DTA [30]</td>
<td>33.33</td>
<td>47.80</td>
</tr>
<tr>
<td>Ours</td>
<td>19.52</td>
<td>33.56</td>
</tr>
</tbody>
</table>

Figure 4: Attack comparison samples. Zoom for detail.

Figure 4: Attack comparison samples. Zoom for detail.

4310

Attack Comparison on Physically-Based Simulation. We run a more extensive attack comparison by using diverse camera poses and evaluated models. Specifically, we follow FCA to evaluate all methods on SSD [21], Faster R-CNN (FrRCNN) [26], and Mask R-CNN (MKRCNN) [14] as the black-box model while keeping YOLOv3 as the target.

The attack comparison results are depicted in Tab. 3, from which we can infer that our method consistently has the best attack performance on all models, both on a single-stage and two-stage detector. Note that we include a random and naive camouflage pattern to show the model’s robustness against arbitrary textures. Again, we can see that DAS and FCA, whose limitation prevents them from considering physical parameters during optimization, yield much lower performance. Also, while CAMOU [45] and ER [41] consider physical parameters from the simulator during optimization, such black-box attacks do not guarantee an optimum result. On the other hand, DTA [30] yields much better results since it accounts for both physical parameters and white-box attacks. Nevertheless, it consistently underperforms compared to ours.

We also provide the model prediction sample of the compared methods in Fig. 4, showing the model can still correctly predict Random, DAS, and FCA textured cars with high detection scores. FCA failure illustrates the importance of considering physical parameters, as the car’s metallic material may cause light reflections that can hide the texture. Naive, CAMOU, and ER decrease the detection score, but still insufficiently to result in misclassification. Alternatively, DTA texture does misclassify the object, but nowhere near our method, which renders the car undetected.

Fig. 5 shows the summarized performance of each camera pose; values are car AP@0.5, averaged from all evaluated models. It visualizes how our method invariably outperforms previous works on all viewing conditions. As implied, our method is relatively stable under various distances compared to other methods. Additionally, we observe that varied camera pitches have a greater impact on attack performance than other poses.
in misclassification, but the other is still detected correctly with a fairly low detection score. It is different from ours, which consistently makes the cars undetected, further verifying the universality of our method as instance-agnostic.

**Transferability Comparison on Different Class.** We evaluate the textures when applied to different classes (i.e., truck and bus), use all evaluated models in the last experiment, and group the result in Fig. 7. As shown, our method constantly outperforms others, demonstrating that ACTIVE is class-agnostic, thanks to our stealth loss that considers all instead of specific class labels during optimization.

**Transferability Comparison on Different Task.** We further evaluate all methods’ universality by testing the generated texture on a publicly available pretrained segmentation model: MaX-DeepLab-L [37] and Axial-DeepLab [38] with SWideRNet [9] Backbone. We test on both Cityscape [10] and COCO [20] pretrained models, and exclude high-pitch cameras because the Cityscape dataset only uses low-pitch data. Furthermore, we only evaluate the pixel accuracy (%) of the car label to show how the texture can downgrade the prediction of the target object. Again, the experiment result in Tab. 5 shows that our method significantly outperforms the previous works. Additionally, Fig. 8 visualizes the sample of how our method makes the car invisible from the segmentation model (either predicted as road or ignored), while other methods correctly predict as car, which also confirms our method produces task-agnostic texture.

**Transferability to Real World.** Following [30], we conduct a real-world evaluation by constructing two 1:10-scaled Tesla Model 3s with a 3D printer and wrapping the texture onto the body of the car: one for a normal and another for our camouflaged car targeting YOLOv3. Fig. 9 shows the normal car model is well detected, whereas the adversarial camouflage car model is not detected as a car at all. Furthermore, we also evaluate practical real-time ob-
Table 5: Universality evaluation on a different task (i.e., Segmentation Model). Values are pixel accuracy (%) of car.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Cityscape Prerained</th>
<th>COCO Prerained</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MaX-DL-L</td>
<td>MaX-DL-L</td>
</tr>
<tr>
<td>Normal</td>
<td>90.70</td>
<td>92.76</td>
</tr>
<tr>
<td>Random</td>
<td>78.20</td>
<td>88.24</td>
</tr>
<tr>
<td>Naive Cam.</td>
<td>50.23</td>
<td>74.78</td>
</tr>
<tr>
<td>UPC [16]</td>
<td>74.70</td>
<td>83.66</td>
</tr>
<tr>
<td>CAMOU [45]</td>
<td>62.12</td>
<td>71.66</td>
</tr>
<tr>
<td>ER [41]</td>
<td>71.55</td>
<td>85.70</td>
</tr>
<tr>
<td>DTA [30]</td>
<td>31.53</td>
<td>55.68</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>17.45</strong></td>
<td><strong>32.04</strong></td>
</tr>
</tbody>
</table>

Table 6: AP@0.5 (%) of target car in real-world evaluation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Real-time Object Detector</th>
<th>YLv3</th>
<th>MbNetv2</th>
<th>EfDetD2</th>
<th>YLX-L</th>
<th>YLv7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>90.83</td>
<td>80.83</td>
<td>96.25</td>
<td>96.25</td>
<td>95.00</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td><strong>8.75</strong></td>
<td><strong>26.27</strong></td>
<td><strong>26.25</strong></td>
<td><strong>40.41</strong></td>
<td><strong>48.75</strong></td>
<td></td>
</tr>
</tbody>
</table>

4.4. Ablation Study

Impact of Proposal on Performance and Naturalness. We evaluated our proposed components, including modules and losses, using ablation studies with default parameters. We used DTA [30] as a baseline since our approach has a similar pipeline. The results in Table 7 demonstrate that each of our proposed components plays a crucial role in enhancing the attack performance. Specifically, utilizing both TPM and ROA modules significantly impacts performance enhancement, with an average improvement of 35%.

Although our proposed losses have a lesser impact on performance, they play an important role in texture naturalness, leading to trade-offs between the two. As illustrated in Fig. 10, omitting $L_{sm}$ outputs a rough texture, whereas excluding $L_{cm}$ makes it more colorful and bright. Employing the stealth loss with only smooth loss yields the best performance, downgrading the car AP@0.5 to 20.21%. More details are available in Supplementary Materials.

5. Discussions

Societal Implications. Adversarial camouflage poses harmful repercussions for self-driving cars since there exists a highly feasible attack scenario, e.g., legally painting cars with adversarial texture [45, 30]. Enhancements in robustness and universality by ACTIVE can amplify the danger as existing public detectors are still highly vulnerable, signifying the importance of research in model robustness.

Limitation. Even though ACTIVE produces a more natural pattern similar to graffiti, the texture is still abstract.

6. Conclusion

We have presented ACTIVE, a physical camouflage attack framework for 3D objects for enhanced universality and robustness. Verified in our comprehensive evaluations, ACTIVE surpasses the performance of existing works—and notably, demonstrates its capability as a model, instance, class, task, and domain-agnostic framework.

Acknowledgments. This work was supported by the Agency For Defense Development Grant Funded by the Korean Government (UE221150WD), and by the MSIT (Ministry of Science and ICT), Korea, under the Convergence security core talent training business (Pusan National University) support program (IITP-2023-2022-0-01201) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Figure 8: Transferability to segmentation model. Zoom in.

Figure 9: Real-world evaluation using two scaled cars. The upper row is the normal car model, while the bottom row is the adversarial camouflaged car model. Zoom for detail.

Figure 10: Visual textures comparison with different losses.
References


[12] Epic Games. Unreal engine. 5


Donghua Wang, Wen Yao, Tingsong Jiang, Guijiang Tang, and Xiaoqian Chen. A survey on physical adversarial attack in computer vision, 2022.

