Physical Attack on Monocular Depth Estimation with Optimal Adversarial Patches

Zhiyuan Cheng[1], James Liang[2], Hongjun Choi[1], Guanhong Tao[1], Zhiwen Cao[1], Dongfang Liu[2,*], and Xiangyu Zhang[1,*]

[1] Purdue University
[2] Rochester Institute of Technology
{cheng443, choi293, taog, cao270, xyzhang}@cs.purdue.edu
{jcl3689, dongfang.liu}@rit.edu

Appendix

This document provides more details about our work and additional experimental settings and results. We organize the content of Appendix as follows:

- §A: Optimizing multiple patch regions.
- §B: Style transfer loss terms.
- §C: MDE model selection criteria.
- §D: Transferability evaluation.
- §E: More ablation studies.
- §F: Physical world experiments settings.
- §G: 3D object detection settings.
- §H: Defence methods and discussion.

A Optimizing multiple patch regions

The patch region does not have to be one part. There could be multiple patches on the target object and our regional optimization technique also supports optimizing multiple patches. In this case, the final patch mask is the sum of multiple sub-masks with each sub-mask representing one rectangular region. The final mask is defined in Equation A1, where $m_p^{\Theta_i}$ refers to the i-th sub-mask and Θ_i denotes its boundary parameters and $clamp()$ is a function to restrict the mask values in between 0 and 1. The mask loss $\mathcal{L}_m$ is also the sum of all sub-mask loss terms.

$$m_p^k = clamp(\sum_{i=0}^{k} m_p^{\Theta_i}, 0, 1) \tag{A1}$$

Fig. A1 gives an example of the optimization process of 3×3 initial patches. As the optimization continues, some patches are minimized and disappear, and

* Corresponding authors

the final patch is dominated by a single one when the target ratio is 1/9 of the vehicle's back view. We test optimizing with different initial patch setups and the result is shown in Fig. A2. Each curve represents the change of attack effect as the total patch ratio decreases and $i \times j$ refers to a initial setup of i rows and j columns. As shown, when the target mask ratio is 1/9, different initial patch setups have similar attack performance at the end with the same total patch area. Thus, we mainly focus on optimization of a single patch (*i.e.*, 1×1 setup) in our evaluation.

B Style transfer loss terms

Style Loss. Let F be the feature extraction network, which can be a pre-trained CNN model, x_s be the style reference image, x' be the adversarial patch example that we will update iteratively. The style loss is defined as the style distance between target image and adversarial example:

$$\mathcal{L}_s = \sum_{l=1}^{L} \|G(F_l(x_s)) - G(F_l(x'))\|_2^2 \tag{A2}$$

, where F_l is the extracted features at the l-th layer of F and G is the Gram matrix of the deep features. L is the total number of convolutional layers in F.

Content Loss. The content loss is designed to preserve the content of the original image since the style loss could make the adversarial example different a lot from the original one. It is defined in Equation A3:

$$\mathcal{L}_c = \sum_{l=1}^{L} \|F_l(x) - F_l(x')\|_2^2 \tag{A3}$$

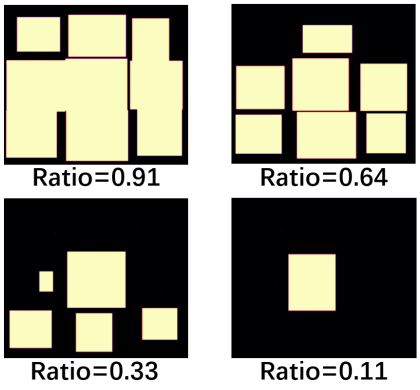


Fig. A1: Optimization with multiple initial patches.

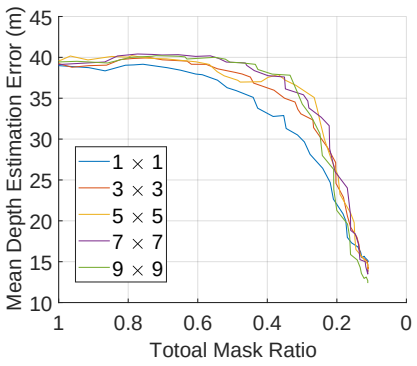


Fig. A2: Optimization with multiple initial patches.

, where x is the original image (*i.e.*, content image). The content loss is to make sure the adversarial example and the original image have similar representation in the deep feature space. In the beginning, the adversarial example is initialized as the content image and the content loss is zero. Content loss will increase once the adversarial example is updated.

Photorealism Regularization. This term introduced in [9] is to constrain the reconstructed image (*i.e.*, adversarial example) to be represented by locally affine color transformations of the content image to prevent distortions [9], which could make the generated image more realistic. Formally, it is built upon the Matting Laplacian by Levin et al. [8] and defined as follows:

$$\mathcal{L}_r = \sum_{c=1}^{3} V_c(x')^\top \mathcal{M}(x) V_c(x') \tag{A4}$$

, where c represents the c-th color channel and $V_c(x')$ outputs the vectorized version of the c-th channel of the adversarial example (*i.e.*, $V_c(x') \in \mathbf{R}^{N\times 1}$, where N is the number of pixels in image x'). $\mathcal{M}(x) \in \mathbf{R}^{N\times N}$ is a matrix only depending on content image x and it represents a standard linear system that can minimize a least-square penalty function described in [8]. We refer readers to the original article for a detailed derivation.

Smoothness loss. This loss is designed to reduce the difference between adjacent pixels and encourage a locally smooth output image. As pointed out in [14], the smoothness term is useful in improving the robustness of a physical-world adversarial examples. The smoothness loss is defined in Equation A5:

$$\mathcal{L}_t = \sum_{i,j} \left((x'[i,j] - x[i+1,j])^2 + (x'[i,j] - x[i,j+1])^2 \right)^{\frac{1}{2}} \tag{A5}$$

, where $x[i,j]$ represents the pixel at i-th row and j-th column of image x.

C Model selection criteria

MDE models can be either trained with supervised method (using ground truth depth collected by Lidar or depth camera) or unsupervised method (using video frames or stereo image pairs). Unsupervised models are more attractive to industry because one can easily collect a large amount of training data (e.g. videos) with affordable RGB cameras or reuse existing videos at a low cost. Tesla has declared that they use a self-supervised model in monocular depth estimation[5]. Hence, in our evaluation, we use three monocular depth estimation models: Monodepth2 [7], Depthhints [16], and Manydepth [15]. We selected these models with the following criteria.

(1) *Representativeness.* Among self-supervised monocular depth estimation models, these models are most widely used in many previous research [17,12]. Monodepth2 [7] is one of most successful monocular depth estimation methods. Depthhints [16] is an advanced model that improves performance via additional

depth suggestion obtained from stereo algorithms. Manydepth [15] is the state-of-the-art model that uses sequence information from multiple images to achieve better performance.

(2) *Practicality.* We focus on self-supervised monocular depth estimation models because they do not require ground truth depth data for training, which is usually collected by high-priced Lidar sensors. In contrast, they require only monocular videos or stereo pairs collected by RGB camera(s), enabling them to collect data and train a model economically and efficiently. These techniques have already been in production-level vision-based autonomous driving systems such as Tesla Autopliot [5] and Baidu Apollo Lite [1].

(3) *Open Model.* The models are publicly available. In our evaluation, we use models trained with both monocular videos and stereo pairs on KITTI dataset [6] and the resolution of input images are 320×1024. These models are publicly available in their project repository on GitHub [4,2,3]

D Transferability evaluation

We evaluate the transferability of our adversarial patch in two phases: the transferability across objects and the transferability across networks. For objects, we use three types of vehicles as our target objects. They are a black SUV (V-A), a blue sedan (V-B) and a grey truck (V-C). We use Monodepth2 as the target depth estimation model and generate adversarial patch for these three vehicles respectively. Then we paste each generated patch to three vehicles and evaluate the attacking performance. For the transferability across networks, we use the black SUV as target object and generate adversarial patches with three kinds of monocular depth estimation models, then we evaluate the attacking performance of each patch on the three networks respectively. Other experimental setup is the same as the effectiveness evaluation and we report the mean depth estimation error for attacks.

Table A1: Transferability evaluation.

(a) Across Objects

	V-A	V-B	V-C
V-A	**12.84**	7.89	9.66
V-B	9.11	**10.23**	5.37
V-C	6.36	8.72	**11.58**

(b) Across Networks

	Many	DH	Mono
Many	**5.876**	1.926	4.649
DH	0.524	**10.027**	9.037
Mono	0.304	1.9	**12.84**

The result is shown in Table A1. In Table A1a, the first column denotes the object for which the patch is generated and the first raw denotes the object to which the patch is pasted. Observe that the adversarial patch generated from one vehicle is also effective on other vehicles, which means that our adversarial patch has a good transferability across objects. At the same time, the patch optimized for the target object has the best attacking performance compared with unmatched objects, which shows the effectiveness of our object-specific optimization. In Table A1b, the first column denotes the target network used to generate the patch and the first row denotes the network used to evaluate the attack. Results on the diagonal are white-box attacks and others are black-box

attacks. Observe that all three models are vulnerable to white-box attacks, which is consistent with our evaluation of attack effectiveness. For black-box attacks, Monodepth2 is the most vulnerable since patches generated from other two networks have a strong effect on it while Manydepth and Depthhints are more robust. In summary, our attack has a good transferability towards Monodepth2 but is less effective on Depth Hints and Manydepth.

E More ablation studies

Style transfer Weight. We also evaluated the impact of different style transfer weight λ. Using a larger style transfer weight can generate more stealthy adversarial patterns while having worse attack performance. This experiment also shows the trade-off between stealthiness and attack effectiveness. We use our default setting in all other experiments as a reference (*i.e.*, $\lambda = 1$) and do ablation study with different style transfer weights. We use Monodepth2 and the vehicle as our target network and object. For a fair comparison, we fix the patch region to the optimized region generated in our default setting and place the vehicle at the same position on scene images in each test. Table A2 presents the result. The first column is the style transfer weight parameter. The second and third columns show the generated patch image and the corresponding depth gap caused by the attack. The fourth and fifth columns are the two metrics we used to evaluate attack performance. The last column reports the Structural Similarity Index (SSIM) between the adversarial patch and the original style image, which is a metric to measure the perceptual difference between two images, ranging from 0 to 1, the higher, the more similar. As shown, using a larger style transfer weight can generate more stealthy adversarial patterns while having worse attack performance. Our default setting ($\lambda = 1$) makes a good balance between them.

F Physical world experiments settings

In our physical world experiments, we use 2016 BMW X1 as our target object and Monodepth2 as the target monocular depth estimation model. We first take

Table A2: Patches generated with different style transfer weights

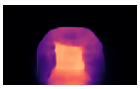
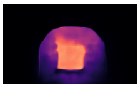
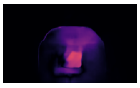
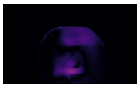
λ	Patch Image	Depth Gap	$\mathcal{E}_d$	$\mathcal{R}_a$	SSIM
0.1			11.09	0.373	0.108
1			7.69	0.246	0.575
10			1.44	0.003	0.924
100			0.65	0.001	0.993

Fig. A3: Physical world experiments setup

a photo of the vehicle's back view. Then we generate the adversarial patch with our attack method as described in §3. Multiple background scenarios from the dataset are used in this generation process. The vehicle with the generated patch is shown in Fig. 9a. We print the patch and paste it on the optimized region of the target vehicle to create an adversarial car. On the victim side, we use iPhone 11's back camera as the main camera of the victim vehicle. We drive the victim vehicle following the target vehicle at a distance of 7-10 meters and record the adversarial scenario while driving. Fig. A3 (a) and (b) show the inside and outside view of our experimental setup. To explore the attack performance under different conditions, we drive on three routes as shown in Fig. A3 (c), which involves different lighting conditions (*e.g.*, positions of the sun and shadows), driving operations (*e.g.*, going straight and turning), and different background scenes and objects. We drive twice on each route, with the first one a benign case and the second one an adversarial case. Specifically, in the first trip, we drive without any patch, while in the second one, we drive with the patch pasted on the target vehicle. We compare the monocular depth estimation of the benign and the adversarial case to evaluate the effect of our patch. Specifically, we report the mean depth estimation error $\mathcal{E}_d$ of the vehicle in both benign and adversarial cases. As we can see in Fig. 4, the depth (z) of the vehicle can be calculated with $z = fH/s$. So, given the focal length (f) of the camera and the height of the vehicle in the physical world (H) and on the image plane (s), we calculate the vehicle's depth. We use this depth as vehicle's depth ground truth to calculate $\mathcal{E}_d$. Also, we project the depth map to pseudo-Lidar point cloud and use PointPillar network to do 3D object detection.

G 3D object detection settings

We use PointPillars as our point cloud-based 3D detection network. The original model is trained with real Lidar data in KITTI object detection dataset. It cannot be directly applied to our pseudo-Lidar data because of their different density and distribution. Hence, we replace the real lidar data in the dataset with corresponding pseudo-Lidar data and train our own model on the new dataset. Specifically, we use Monodepth2 as the monocular depth estimation model and generate pseudo-Lidar for all images in the KITTI object detection dataset replacing original Lidar data. Then we train our PointPillar network on the generated pseudo-Lidar dataset from scratch. The training is the same

as original setup and the mean average precision (mAP) of our model on the category of cars is 61.04, which is close to the performance in Apollo (*i.e.*, 63.49).

In each scene, we place the adversarial vehicle at a distance of 7 meters in the front of the victim vehicle, then we predict the depth of the scenario and project the depth output to 3D space generating pseudo-Lidar point cloud. Next, we use PointPillar to detect 3D objects in this scenario with the point cloud as input. We evaluate on the three depth estimation models and a vehicle object, and use three different target patch sizes in patch region optimization. We perform the evaluation on 100 scenarios and report the attack success rate.

H Defence methods and discussion

In §4.5, we evaluated five popular defencing techniques. The introduction and configuration details of these methods are as follows:

JPEG compression: This method uses JPEG image compressing algorithms to compress the input before feeding to the depth estimation network. The compressing operation is expected to disturb the subtle pixel-level adversarial noise and defend the adversarial attack. In our evaluation, we use Python Image Library (PIL) to apply JPEG compression to the input and select the quality level from 90 to 20. Lower quality argument means higher compression rate.

Bit-Depth Reduction: Typical RGB images have three channels and each channel has an 8-bit depth. Pixel value of each channel ranges from 0 to 255. Bit-depth reduction is to remap the 8-bit depth to a smaller bit depth. Lower bit-depth has smaller color space and this remap operation can also disturb the adversarial perturbation to defence the attack. In our experiments, we evaluated four smaller bit-depth cases from 5 bits to 2 bits.

Median Blur: Median Blur is a method to smooth the image by calculating the median of each pixel's surrounding pixels within a certain kernel size. This defence uses the smoothing effect to remove the adversarial noise. We use the median filter implemented in SciPy and use square kernel size ranging from 5 to 25 in our experiments. Larger kernel size has stronger smoothing effect.

Gaussian Noise: This method adds Gaussian noise on the image to disturb the adversarial perturbation since adversarial perturbation is also a kind of precise noise designed to fool the network. The Gaussian noise we add to the image is zero-mean and the standard deviation is from 0.01 to 0.1. As a reference, the image data is normalized to [0, 1]. Gaussian noise with larger standard deviation is stronger.

Autoencoder: The Autoencoder is a method proposed in Magnet [11] to defend adversarial attacks. It uses neural networks to filter out the adversarial noise which is not within the distribution of training dataset of the target model. The neural network architecture differs according to dataset and the size of input images, and in our evaluation we use architectures defined in the original work (minist and cifar10) and the architectures designed in [13] (Arch-1 and Arch-2) for large-size images. We train these networks with the KITTI dataset and eliminate the 100 scenes used in our evaluation to avoid in-sample evaluation.

Besides those methods we evaluated above, adversarial training [10] is an effective method to improve the robustness of DNN models against adversarial

examples. However, traditional adversarial training is for supervised learning which requires ground truth data while the depth estimation models we target are trained in an unsupervised (*i.e.*, self-supervised) way using videos and stereo image pairs. Hardening self-supervised MDE models with adversarial training effectively and efficiently is still an open problem and we leave it to future work.

Another direction is to fuse pseudo-lidar and RGB image. Since we consider fully vision-based perception system, the defense cannot include other types of sensors like Lidar, Radar or ultrasonic sensors. To avoid object detection failure, one direction is to make full use of camera frames by fusing pseudo-Lidar and RGB images. Although the point cloud of the target object is distorted by the adversarial patch, the object can still be detected in the RGB image. Fusing both sources may have more robust object detection result. Note that this cannot fundamentally defeat our attack because the object detected in the RGB image does not have depth information and the spatial relationship between the detected target object and the victim vehicle can still be wrong. Also, fusion does not solve the problem of wrong depth estimation.

References

1. Baidu unveils Apollo Lite Level 4 vision-based autonomous driving solution, https://autonews.gasgoo.com/m/Detail/70016068.html
2. Depth Hints Github, https://github.com/nianticlabs/depth-hints
3. Manydepth Github, https://github.com/nianticlabs/manydepth
4. Monodepth2 Github, https://github.com/nianticlabs/monodepth2
5. Tesla use per-pixel depth estimation with self-supervised learning, https://youtu.be/hx7BXih7zx8?t=1334
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
7. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth prediction (October 2019)
8. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. IEEE transactions on pattern analysis and machine intelligence **30**(2), 228–242 (2007)
9. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4990–4998 (2017)
10. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
11. Meng, D., Chen, H.: Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 135–147 (2017)
12. Ramamonjisoa, M., Firman, M., Watson, J., Lepetit, V., Turmukhambetov, D.: Single image depth prediction with wavelet decomposition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (June 2021)
13. Sato, T., Shen, J., Wang, N., Jia, Y., Lin, X., Chen, Q.A.: Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In: 30th {USENIX} Security Symposium ({USENIX} Security 21). pp. 3309–3326 (2021)

14. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: 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. pp. 1528–1540 (2016)
15. Watson, J., Aodha, O.M., Prisacariu, V., Brostow, G., Firman, M.: The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In: Computer Vision and Pattern Recognition (CVPR) (2021)
16. Watson, J., Firman, M., Brostow, G.J., Turmukhambetov, D.: Self-supervised monocular depth hints. In: The International Conference on Computer Vision (ICCV) (October 2019)
17. Yang, N., Stumberg, L.v., Wang, R., Cremers, D.: D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1281–1292 (2020)