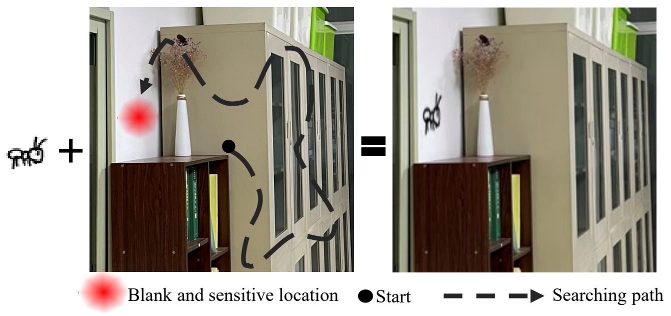
Attck Monocular Depth Estimation with a Sticker

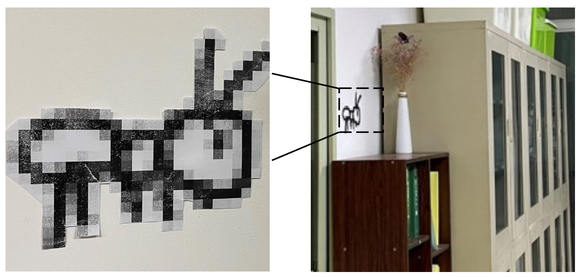
Meng Pan, Fangjun Huang[†]
School of Computer Science and Engineering, Sun Yat-sen University, China
panm9@mail2.sysu.edu.cn, huangfj@mail.sysu.edu.cn

Abstract

DNNs have been demonstrated that their performance can be degraded by imperceptible perturbations. Monocular Depth Estimation networks with DNNs likewise encounter such a dilemma. This attack diagram, adding perturbations, is not implementable and capturable for a sensor, which means it is an ideal attack diagram. An implementable and applicable attack diagram in the physical world is needed. To this end, we address this by sticking patches into indoor scenes. We proposed an automatic searching and transformation method. In the searching phase, the patch is dispatched to an appropriate and sensitive location exploiting the MDE's gradient. In the transformation phase, the patch alternates its appearance according to its location. In doing this, we can generate destructive, inconspicuous adversarial examples. This whole process doesn't rely on any generate networks, which makes it fast and light. We evaluate our method on NYU depth v2, experiments show that our method eliminates 6% of RMSE in 0.8s meanwhile keeping remarkable inconspicuousness.



(a)



(b)

Figure 1. Preview of our patch attack method. The adversarial example generated by our concealed attack method(a). We stick a printed draw in physical world (b), and Note that this was shot in our office, not the dataset scene we used in the experiment part.

1. Introduction

Monocular Depth Estimation (MDE) algorithms is a crucial 3D scene perception method. With the rapid development of Deep Neural Networks(DNNs), many excellent MDE methods base on DNNs have been proposed recently. However, according to Szegedy's research [10], DNNs are prone to be attacked by adversarial examples: when an imperceptible small perturbation is added to an input image, the classifier based on DNNs will classify the obtained image, knowned as adversarial examples, into a wrong category with high probability. Adversarial examples have been found not only on classification tasks but also on logical regression tasks, such as semantic segmentation and object recognition. MDE also fail to get rid of this dilemma. As learned perception modules are increasingly deployed on autonomous driving vehicles, mistaking a stop sign for a speed limit or causing obstacles to disappear can cause immense damage. So as a kind of 3D scene perception solution, MDE algorithms need to be robust and reliable.

The threat of adverserial example on MDE algorithms was first studied by Wong *et al* [12]. They conducted comprehensive and rigorous experiments to explore the impact of pixel attacks on MDE. Including attacking the whole depth map of a scene, attacking the depth map of one single object in the scene, and even making a specific object vanished completely on the depth map by adding perturbations. This research is pioneering and enlightening for attacking MDE.

However, the study of Wong *et al* is hard to be realized in

[†]Corresponding author

the physical world, since the perturbations added on the input images can not be perceived by a camera. However, attacking scenes with patches is an appropriate solution. Camouflaging a patch to a sticker and sticking it into the target scene is inconspicuous and easy to be implemented.

There exist some works exploiting patches to generate adversarial examples on image classification, traffic sign recognition and other tasks. For example, Brown *et al* [1] exploited patches to attack DNNs, but they didn't pay much attention on inconspicuousness, that is, the abrupt and unreal patch in scenes will reveal the attack intention. Later works [3,6,7] noticed this problem and utilize GAN to make patches more naturalistic. But this will increase time and space complexity obviously.

In this paper, we take physical realizability and inconspicuousness into accout, and design a fast and automatic method to attack MDE algorithms. Fig. 1 shows the rough outline of our method. Meanwhile it indicate that our adversarial examples generated on pixel level can be simulated through sticking a printed patch on the real scene.

The contributions of this work are as follows:

- We are the first to apply patch attack on MDE tasks.
- We propose a fast, automatic, inconspicuous attack method. In this method, a learnable matrix is designed to update the patch's location. The updating of the location is based on attack effect and visual quality using gradient descent automatically. Finally we use perspective transform to alter the patch's appearance and make it looks like a sticker.
- experiments show that our method generate destructive while concealed adversarial examples.

.

The structure in the rest paper is organized as follows. After a review of MDE and patch attacks in Sec 2, we introduce our method concretely in Sec 3. Then we visualize the effct of our location searching phase and attack a MDE network in Sec 4. Finally We analyze our patch attack method in Sec 5.

2. Background

In this paper, patch attack for MDE model is concentrated on. We will give a brief introduction for MDE task and review the related patch attack works. Note that we are the first to attack MDE model with patches, so the related works only involved image classification, traffic sign recognition, face recognition, etc. But these works still inspire us a lot.

2.1. Monocular Depth Estimation

MDE algorithms can generate dense depth map for each pixel in the given RGB image. They achieve satisfactory results base on advanced DNNs. Fig. 2 shows a common inference stream of MDE model: A RGB image is fed into the MDE model, which generate dense depth map for the image. The training of the model needs immense, dense annotaed image and depth maps. However, suffered from the lack of sufficient geometric constraints, MDE is still an ill-posed problem. Current MDE networks are bulky and inefficient, which can not be equipped on embedding systems. Wofk et al. [11] proposed Fastdepth, a light, fast MDE network.

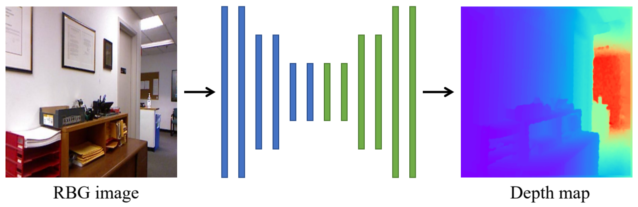


Figure 2. A common stream of MDE algorithm.

2.1.1 Unsupervised

2.2. Patch Attack

Adding perturbations to images has a limit: the imperceptive noise can not be captured by camera. Recently, researchers turn to generating adversarial examples with patches. It can be captured by camera and it can implement to physical world. The only question is, how to make it concealed.

Sharif et al. [9] attacked FRS using real printed glass frames, with which the FRS can not recognize the person or recognize him as someone else in the system. The pattern of these frames are generated by optimizing a composite objective function, including gradient decenting, smoothness of perturbations and non-printability score (NPS) optimal terms. Tom et al. [1] abandon the camouflage of the attack, they attack DNNs with an apparent, universal patch, which is updated with different locations and transformations settings. To generate robust adversaries, eykholt et al. [2] sampled from both actual physical and synthetic transformation images. Besides, they found that there exist vulnerable and robust areas in an image and identified these areas by computing perturbations using the L_1 regularization. Then they generate perturbations using L_2 regularization on vulnerable areas, finally reshape the perturbations to a graffiti shape. In doing so, they achieve a high successful rate under various environmental conditions. Liu et al. [7] also attacked the traffic signs. Instead of locating the vulnerable areas by computing, they locate the patch by a attention model, which takes as input the traffic sign, outputs an mask that indicates where to put the patch. The patch is generated through a GAN, which consists of a generator to generate a

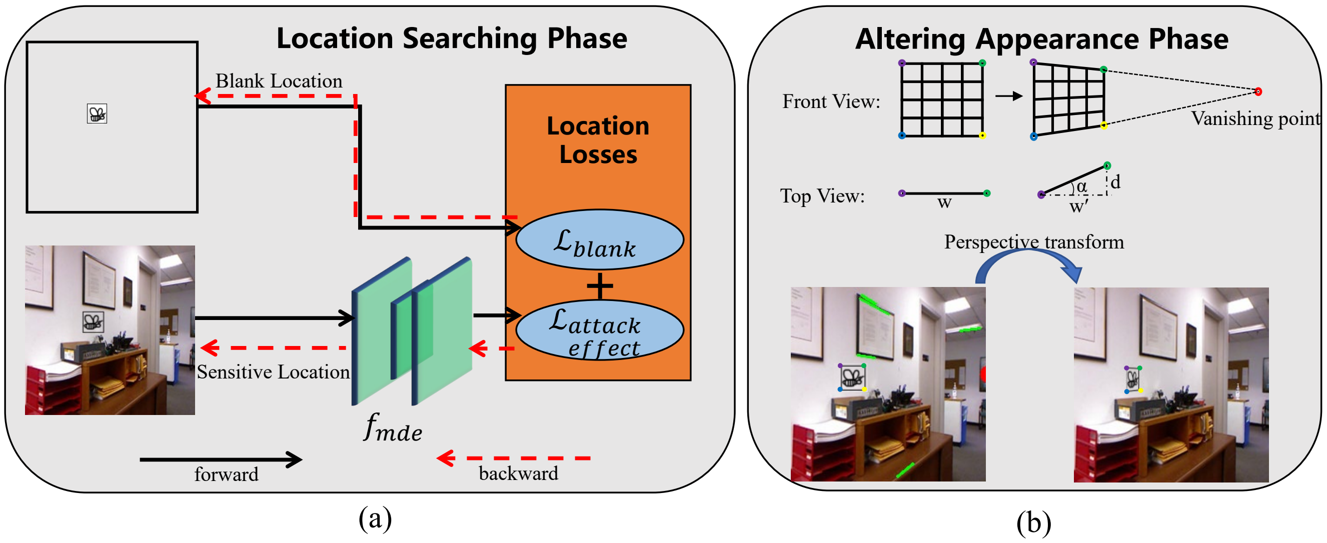


Figure 3. Stream of our MDE attacking method. The whole stream consists of two phases: (a) location searching phase and (b) altering appearance phase. They are described concretely in Sec 3.2 and Sec 3.3

misclassification patch and a discriminator to discriminate whether the patch is original or adversarial. Kong et al. [6] also use a GAN to generate perturbation patches. In particular, they manipulate advertisement posters as original patch, which is artificially stuck on a street billboard after it through a GAN. This method produces better inconspicuousness yet the process of sticking the patch to billboard is labor consuming. In [3], the locations of patches are artificially determined and fixed to make a better camouflage. They first detect all humans in an image and pin a generated patch on their shirt. This attack makes the adversarial examples look as natural as a natural shirt pattern. Forward-mentioned attack methods need to access to the objects they attacked. The advantage is that adversarial examples are more inconspicuous, it is hard to deployment in the physical world though. To this end, Zolfi et al. [13] proposed a contactless attack method. Several translucent patches are generated by updating the affine transformation matrix, then they are deployed on the camera's lens.

3. Method

In this section, we are going to present our MDE attacking method in detail. we first formulize the problem in Sec 3.1. Then we describe how we design our patch, the objective function and how the patches are optimized in Sec 3.2. Finally we present the appearance altering phase in Sec 3.3.

3.1. Problem Statement

Given a MDE data distribution: $\mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H\times W}$, a MDE model f_{mde} can learn a map that simulate the distribution within an error range. Considering elaborately designed perturbations can hardly be detected by consumer cameras, we are going to attack f_{mde} by sticking inconspicuous patternsδ on appropriate areas.

$$I^{adv} = (1-m)\odot I + \delta \tag{1}$$

where$I$ is clean original image, I^{adv} is adversarial example, m is a full-one mask that has the same size as the patch δ. Our goal, attacking a MDE system, is to maximize the error of ground truth depth map D^* and depth map generated from I^{adv}. $\|\cdot\|_p$ denotes the l_p norm.

$$\max \|f_{mde}(I^{adv}) - D^*\|_p \tag{2}$$

3.2. Location Searching

In this subsection we are goint to explicate our location searching phase. Including patch design, our objective function and optimization of the location. The stream of the location searching phase can be seen in Fig 3(a).

3.2.1 Patch Design

To obtain an adversarial example I^{adv}, the first problem we encountered is where to stick the patches. Exhaustive search is feasible but time consuming. We expect to complete this this process automatically and quickly. So we turn to optimizing the patch location by gradient decenting. We define a 2×3 matrix $\mathbf{A}_\theta$ for each patch to implement this location search process. By optimizing this $\mathbf{A}_\theta$, a patch can update its location automatically.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \tag{3}$$

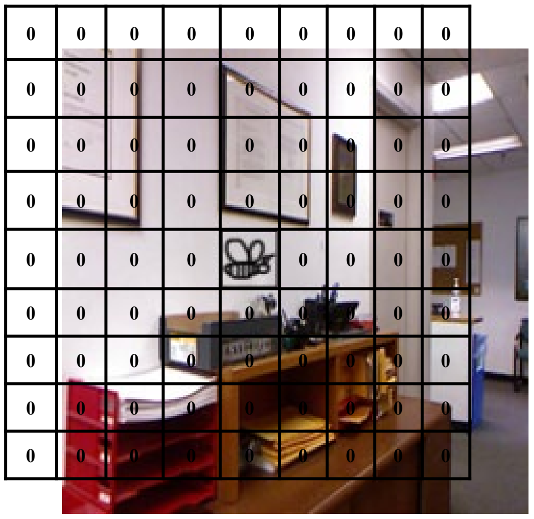


Figure 4. Illustration of our mask M, with which we can move a patch on images to be attacked through $(1-M)\odot I+\delta$.

where $[t_x, t_y]^\top$ denote the offsets $\mathbf{T}$ in horizontal and vertical directions, respectively. Initial values are $[0,0]$, which represent that the center of the patch is on the center of the image.

Specifically, we first place the patch δ in the center of the clean image I, then pad 0 around it to the size of I. In doing so, we got a mask M. The homogeneous coordinate of this mask is noted as G and its size is $H\times W\times 3$. For each pixel of m, $G_i = (x_i, y_i, 1)$. This movement of the mask can be represented as:

$$G'^T = A_\theta\, G^T \tag{4}$$

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{pmatrix} x_t \\ y_t \\ 1 \end{pmatrix} \tag{5}$$

Finally we got the patch movement on the image through moving mask M: $(1-M)\odot I+\delta$.

3.2.2 Objective Function

The aim of this research is to attack MDE models inconspicuously with patches, we consider three factors: attack effect, which area of the adversarial example is more inconspicuous and what appearance of the patches are more inconspicuous. The last consideration about the appearance is addressed by perspective geometry, while the attack effect and inconspicuous areas are optimized by two loss funtions respectively. The two loss functions consisit our objective function in a linear manner:

$$\mathcal{L}_{total} = \mathcal{L}_{attack} + \mathcal{L}_{blank} \tag{6}$$

We are going to explain the two loss functions in the following part of this section.

attack effect MDE f_{mde} systems generate depth maps, whose pixel values represent the distance from area to sample sensors. To obtain best attack effect, that is, to maximize the error between prediction maps and annotated maps:

$$\mathcal{L}_{attack} = \|f_{mde}(I^{adv}) - D^*\|_p \tag{7}$$

We use L_1, L_2 and Huber loss in our experiments.

Visualization Quality For better inconspicuousness, we camouflage the patch as graffiti, which can be usually found on blank walls. Blank means with no pictures, marks or decoration on white walls. Based on this we design Blank loss to help the patch to find blank areas, the Blank loss consisits of two optimization sub-terms:

$$\mathcal{L}_{blank} = \mathcal{L}_{smooth} + \mathcal{L}_{white} \tag{8}$$

$\mathcal{L}_{smooth}$ constrain the patch locate on areas with low texture complexity:

$$\mathcal{L}_{smooth} = -\frac{\sqrt{\partial_x(m\odot I) + \partial_y(m\odot I)}}{n} \tag{9}$$

where ∂_x and ∂_y denote the gradient in x and y directions respectively. n is the number of pixels in m. $\mathcal{L}_{white}$ constrain the patch locate on white areas:

$$\mathcal{L}_{white} = \frac{\sum_1^n m\odot I}{n} \tag{10}$$

3.2.3 location Optimization

As indicated in Eq.5, the coordinates of the target pixels G'_i are continuous values. We adopt bilinear interpolation to determine the value of the target pixel. Bilinear interpolation mechanism is proved to be differentiable in *spatial transformer networks* [5]. By maximizing the $\mathcal{L}_{attack}$, we can optimize the offsets $\mathbf{T}$ step by step.

$$\mathbf{T}_0 = [0,0]^\top, \mathbf{T}_{S+1} = Clip\{\mathbf{T}_S + \alpha \bigtriangledown_{\mathbf{T}} \mathcal{L}_{total}\}. \tag{11}$$

To avoid patches overflowing from RGB images, $\mathbf{T}$ are clipped between $(-0.875, 0.875)$. For all experiments we set $\alpha = 0.004$, total optimization step is 50 steps.

3.3. Appearance Altering

The location problem is settled by optimizing an affine transform matrix $\mathbf{A}_\theta$, next we address altering the appearance of the patch. The appearance altering phase can be seen in Fig 3(b).

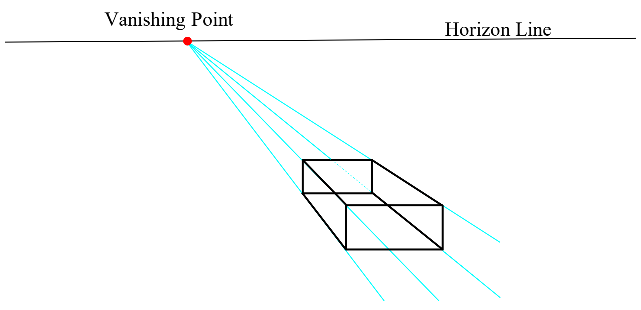


Figure 5. Diagram of single point perspective drawing: All the vanishing lines lead to the vanishing point. The horizontal and vertical lines, however, remain parallel to each other.

When we alter the appearance of the patch, for better visualization quality, we followed the principle of perspective drawing: the vanishing point is where all parallel lines intersect and is always on the horizon line. Fig 5 clearly illustrates this principle.

We first detect straight lines in the attacked scene with *Hough* transform, then we obtain the vanishing point by extending this lines and computing their intersection. Based on the principle of perspective drawing, as illustrated in Fig 3(b), if we connect the left-top point and vanishing point, the right-top point should lay on this line. Same on the bottom line. Meanwhile, the connection lines, which converge to vanishing points, will look shorter than their original length. We explain this fact in Fig 3(b): when seen from the front view, the width of patches w will shrink to w'.

$$w' = \sqrt{w^2 - d^2} \tag{12}$$

Since this task we concentrate on is just MDE, we possess the depth of each pixel. The patches' depth d can be got by subtracting fronter boarder depth from back boarder depth. We set $w = 0.1m$ in our experiments. We get four corrected corner coordinates, with which we alter the appearance of the patch. The perspective transforms in this phase are implemented with *OpenCV*.

4. Experiments

In this section, we conduct several experiments to exhibit the results of our automatic and inconspicuous attack method on the fastdepth [11] MDE system. We introduce our implemention detail, including datasets, data pre-processing and experiment environment in Sec 4.1. In Sec 4.3 and Sec 4.4, our attack effects are shown by visualization and quantitative comparison. Then we demonstrate the automatic searching process driven by our method in Sec 4.2. Finally we show and analyze the visual quality of our adversarial examples.

4.1. Implemention Detail

For the MDE system, we use Fastdepth [11]. We also use the same dataset NYU Depth v2 [8] used in the Fastdepth [11]. This dataset is an indoor dataset, whose depth range is 0-10 m. It consisits of 1449 densely annotated depth and RGB images pairs, 795 of which are split to train dataset and the rest are split to test dataset. For the image pre-processing, we followed the process in Fastdepth: images are resize to 250×333 firstly, secondly, center cropped to 224×304, in the end they are resized to 224×224.

For the patch dataset, we choose QuickDraw [4]. QuickDraw is a collection of 50 million hand-writing drawings across 345 categories. We select several categories, including bee, ant, dolphin, crocodile and so on. The size of them are all 28×28.

We focus on four metrics used in previews works to evaluate the attack effect:

$$RMSE = \sqrt{\frac{1}{|N|}\sum_{i\in N}||d_i - d_i^*||^2} \tag{13}$$

$$MAE = \frac{1}{|N|}\sum_{i\in N}|d_i - d_i^*| \tag{14}$$

$$Abs\ Rel = \frac{1}{|N|}\sum_{i\in N}\frac{|d_i - d_i^*|}{d_i^*} \tag{15}$$

$$Threshold : \%\ of\ d_i\ s.t. \max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < thr \tag{16}$$

thr denotes three threshold 1.25, 1.25^2, 1.25^3. Besides the accuracy, we also record our runtime, which can show the efficiency of our attack method.

4.2. Automatic Searching

In this section you are going to find that given an image our attack method can search the sensitive zone and stick the patch on the zone. To better demonstrate our search effect, we calculate the MAE maps of test images. The MAE maps are obtained using following step: beginning from the left top corner, where is (-0.875,-0.875), we use strides 7 to move the patch step by step and calculate the MAE of each position. The red and violet colors in the maps represent higher and lower MAE values, respectively. As you can see from Fig 6, our method can accurately search the sensitive area of the image. Note that to emphasize our search effect, we didn't use the Blank (eq 8) loss function in this experiment.

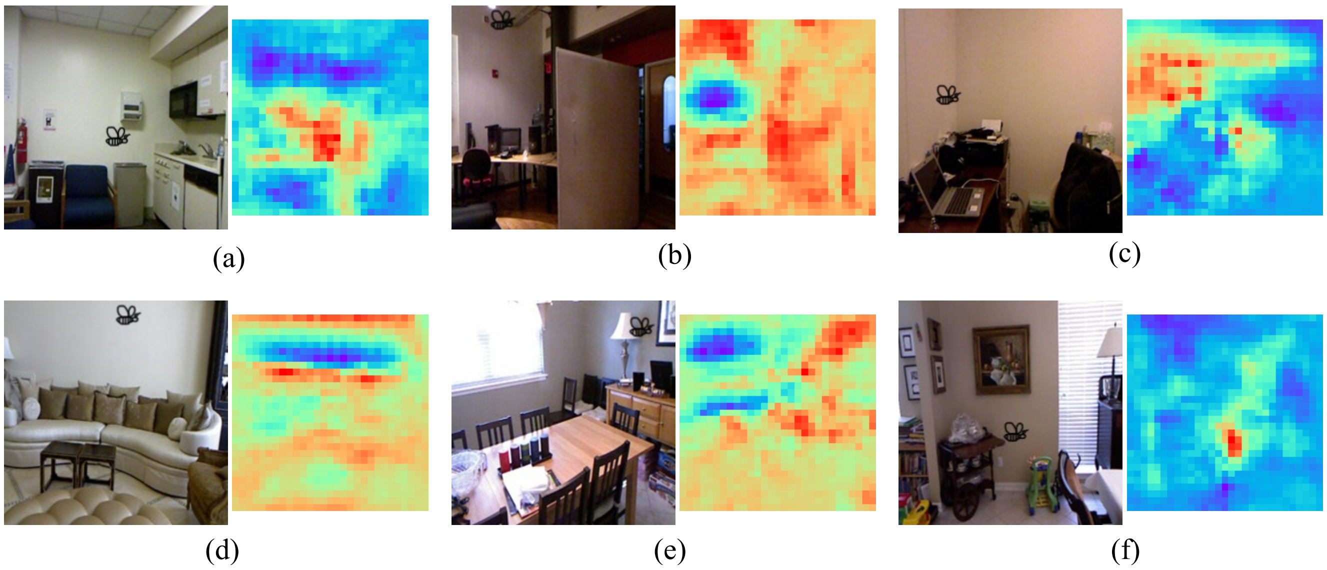


Figure 6. The final attack locations searched by our method (left) and corresponding MAE maps (right). Then calculate MAE maps, the offsets of the patch are clamped between -0.875 and 0.875, which makes the maps a little smaller than the RGB images.

Table 1. Accuracy of our method, best results are boldfaced

		RMSE	MAE	Absrel	$\delta_1 < 1.25$	$\delta_2 < 1.25^2$	$\delta_3 < 1.25^3$	Runtime
fastdepth (baseline)		0.600	0.427	0.162	0.771	0.927	0.988	-
Ours	L_1	0.643	0.464	0.175	0.738	0.920	0.973	0.772
	L_2	0.641	0.461	**0.177/9.3%**	0.739	0.921	0.973	0.781
	Huber	**0.648/6%**	**0.466/9.1%**	0.176	**0.737**	**0.919**	**0.972**	0.772

4.3. Visualization Attack Results

As mentioned above, our method generate inconspicuous and destructive I^{adv}. The results are shown in Fig 7 from the two aspects.

For inconspicuousness, we show the adversarial examples in the first and third rows. The black box shows that our adversarial examples follow the principle of perspective drawing. Besides the patches lay on walls and furniture, which makes our examples look like graffiti and more harmonious in indoor scenes.

For destructiveness, we show the prediction error in the sencond and fourth row. Our method achieves $320mm$ max error. Prediction results from I^{adv} are usually nearer than ground truth depth maps in far-range areas, for instance, the blue areas in Fig 7. However, the results are farther in near-range areas, which means our adversarial examples mislead the MDE system and make the prediction results converge to middle distances.

4.4. Quantitative Attack Results

We show our attack effect by quantitative comparison in Table 1. To the best our knowldege, we are the first to attack MDE systems using patches. There does not exist much related works. So we compare the baseline MDE method, fastdepth between our method with different $\mathcal{L}_{attack}$. We degrade the MAE by 9.13%, as a contrast, Wong [12] achieve 10% accuracy loss (closer or farther) on each pixel of the depth. However, they rely on the guidance of the modified ground truth depth maps, in other words, they modify the ground truth depth value down or up 10% in each pixel, and use the modified maps as supervision. Besides, they use elaborately designed perturbations to generate adversarial examples.

5. Conclusion

We attacked an MDE system using our fast, physical-realizable, automatic and inconspicuous patch attack method. Fast because it doesn't lean on any generative networks. Physical-realizable because the effect can be simulated by sticking a real patch on the sensitive spot. Automatic because it is optimized by gradient descent. Inconspicuous because our method conceals the patch as graffiti. Our experiments showed that the depth maps predicted by the target MDE system lost some accuracy. Our work re-

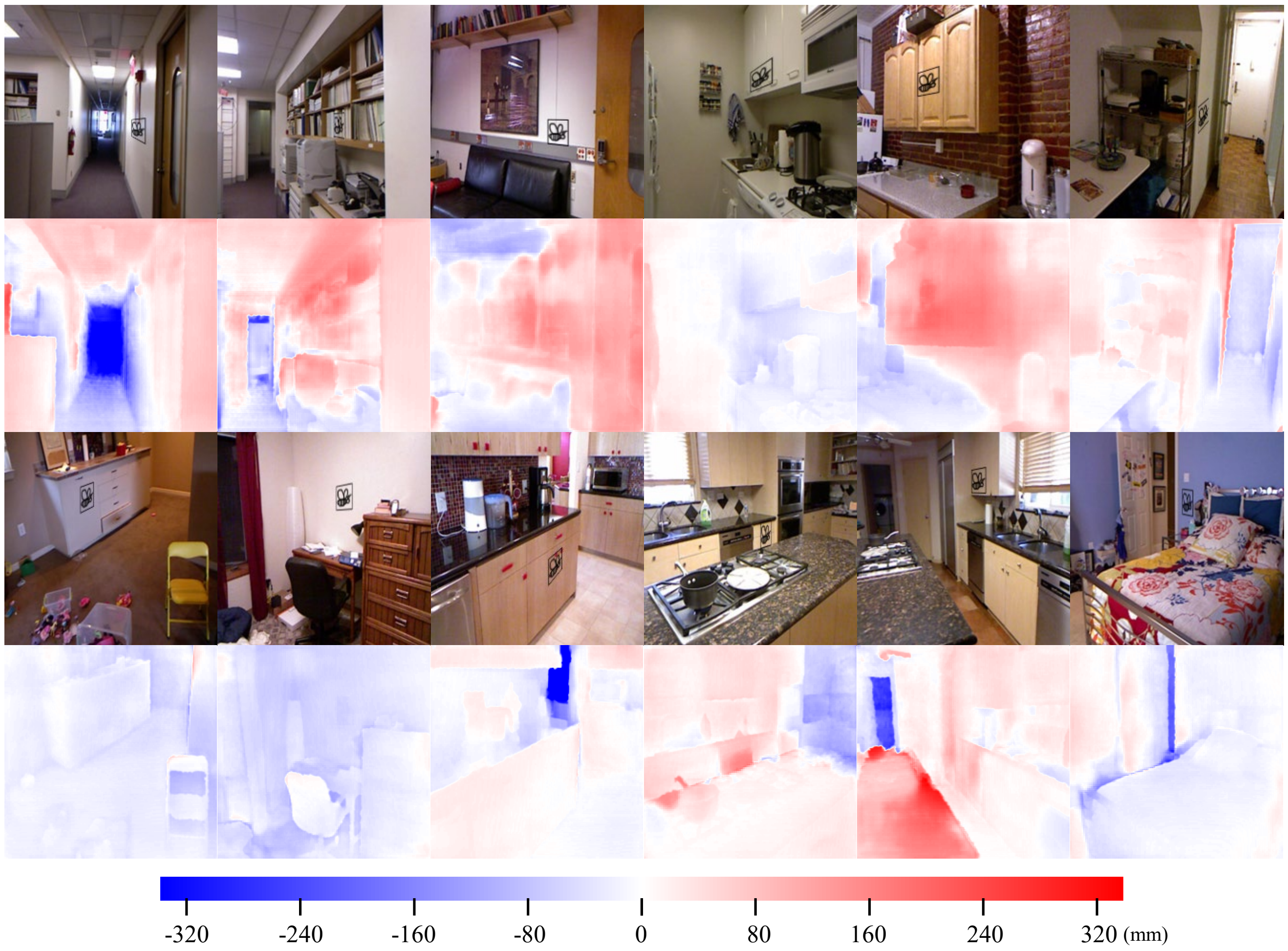


Figure 7. The visualization of our patch attack method. The first and third rows are I^{adv}. The second and fourth rows are the visualization of prediction error: $f_{mde}(I^{adv}) - D^*$. The color bar down the results explicates the error value intuitively. We emphasize the patches using surrounding black boxes, which are transformed in the altering appearance phase also.

vealed the probability that an accident could happen due to drawing graffiti intentionally or unintentionally. Future works should pay more attention to multimodal fusion or provide robust constraints for MDE networks.

6. Acknowledgements

xxx

References

[1] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[2] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[3] Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7848–7857, October 2021. 2, 3

[4] T. K. J. K. J. Jongejan, H. Rowley and Fox-Gieg. The quick, draw!, 2016. `https://github.com/googlecreativelab/quickdraw-dataset`. 5

[5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2017–2025, Cambridge, MA, USA, 2015. MIT Press. 4

[6] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for

autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3

[7] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. 2

[8] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[9] 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. 2

[10] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 1

[11] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108, 2019. 2, 5

[12] Alex Wong, Safa Cicek, and Stefano Soatto. Targeted adversarial perturbations for monocular depth prediction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8486–8497. Curran Associates, Inc., 2020. 1, 6

[13] Alon Zolfi, Moshe Kravchik, Yuval Elovici, and Asaf Shabtai. The translucent patch: A physical and universal attack on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15232–15241, June 2021. 3