Adversarial Patch for 3D Local Feature Extractor

Pao, Yu-Wen
National Taiwan University
Taipei, Taiwan
b09902016

Lin, Hong-Yi
National Taiwan University
Taipei, Taiwan
b09902100

Li Chang Lai
National Taiwan University
Taipei, Taiwan
b09902135

Abstract

Local feature extractors are the cornerstone of many computer vision tasks. However, their vulnerability to adversarial attacks can significantly compromise their effectiveness. This paper discusses approaches to attack sophisticated local feature extraction algorithms and models to achieve two distinct goals: (1) forcing a match between originally non-matching image regions, and (2) preventing a match between originally matching regions. At the end of the paper, we discuss the performance and drawbacks of different patch generation methods.

1 Introduction

Local feature extractors have become the backbone of many computer vision tasks that have revolutionized our world. Self-driving cars, for instance, rely heavily on accurate feature extraction to navigate safely. However, what if these powerful models misinterpret what they see?

This paper explores a specific adversarial attack that exploits how deep learning models interpret visual information. Generally, these models rely on local feature extractors to detect tiny snippets of an image, like edges or textures, to make sense of the bigger picture. This research paper examines how generating minor adjustments to an image can lead the model to misinterpret a scene.

Imagine a self-driving car encountering a stop sign. The car's computer vision model identifies the red octagon with local features. Our approach involves placing two small patches on the sign that appear different depending on the angle you look from. By confusing the local features, we hope to show how the model might misinterpret the entire scene, potentially with disastrous results.

2 Related works

2.1 Local feature extraction

The local feature extraction is to describe the image based on each local area of the image. The local feature extraction usually comes with two stages. The first stage, also known as feature detection, is to locate a set of points, objects, or regions in the images. The second stage is to create a descriptor for each feature point. In this work, we concentrate on SuperPoint[3], a local feature extractor based on deep learning. The SuperPoint is a CNN-based model. The input will first passed into the encoder to encode a shared representation for the interest point decoder and the descriptor decoder. The interest point decoder can be seen as a classifier to find the position of the feature point for each non-overlapped 8×8 region. The descriptor decoder gives the 256 channel descriptions for each region.

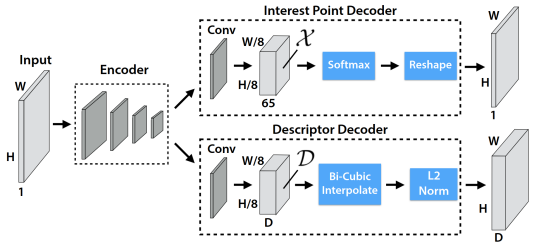

Figure 1: The model architecture of the SuperPoint[3]

2.2 Projective transformation

Projective transformation[6], also known as the homography, describes the change of the perceived object when the viewpoint changed by a 3×3 matrix, H, which is a homogeneous matrix.

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

To be more specific, for a point, (x, y), transform to a point (x', y') when changing to the new viewpoint by applying a homography H. It will be:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

3 Methods

3.1 Overview

There will be two adversarial patches in the same scene. We denote the source patch as P_{source}. The other one, the target patch, is denoted as P_{target}. For the different viewpoints, the source view, V_{source}, and the target view, V_{target}, we want to increase the number of mismatches between the source patch at the source view and the target patch from the target view. The higher the mismatch rate is, the more likely it is to fail the downstream tasks. The proposed attack is composed of two parts. One is to generate an adversarial patch that the local feature extractor is sensitive to, while the other part is to determine the mask to which the adversarial patch will be applied.

3.2 Adversarial patch generation

The baseline adversarial patch is the chessboard pattern. Due to the local feature extraction design, every junction point between four blocks on the chessboard should be identified as a local feature point. What's more, the targeted local feature extractor, SuperPoint[3], uses synthetic data similar to the chessboard as the input of the pre-training. Hence, the SuperPoint is sensitive to chessboard patterns naturally. We use $8 * 8$ size for each small cell in the chessboard pattern.

Besides of handcraft pattern, we want to generate a pattern that SuperPoint is sensitive to based on its model weights directly. Inspired by FGSM[5] and PGD[9], we create the adversarial patch, x, by multiple steps of gradient ascent by the following formula:

$$x^t = x^{t-1} + \alpha \nabla_{x^{t-1}} L$$

where α can be seen as the learning rate, L is the loss function at t step. Since the interest point detector is a classifier, we can design two scenarios, one is the targeted class and the other is the untargeted class. Their loss functions will respectively be:

$$L_{ce}(\theta, x, y_{target}) \text{ and } -L_{ce}(\theta, x, y_{dustbin})$$

where L_{ce} is the cross-entropy loss, θ is the model weight, y_{target} can be any position in a 8×8 patch, and $y_{dustbin}$ indicates the class that there's no local feature in the area.

Based on the early experimental results in 4.3, we found that the inconsistent size of the patch and the mask may cause a decrease in the performance. Hence, we add augmentation like resizing and random cropping to increase the ability of the scale-invariant of the adversarial patch.

However, most of the performance of the chessboard pattern is better than the adversarial patch based on the experimental results 4. Hence, we try to directly inherit the performance of the chessboard and further boost the performance of the attack. Instead of the random noisy or gray-scale image, we use the chessboard as the initial image for the optimization. And then, apply the update with the augmentation.

chessboard targeted adv. untargeted adv. aug. chess-init

Figure 2: The adversarial patches.

3.3 Mask generation

Mask generation determine the position and the shape, P_{source} and P_{target}, that the adversarial patch will be filled in. Since the intuition is to increase the similarity between P_{source} at V_{source}, and P_{target} at V_{target}, P_{target} at V_{source} should be similar to P_{source} at V_{srouce} after applied the homography transformation matrix, H, from V_{source} to V_{target}. Let's simply denote P_{source} at V_{source}, as P_{source}, P_{target} at V_{source}, as P_{target}, and P_{target} at V_{target}, P'_{target}.

$$P_{source} \sim P'_{target}$$

$$P_{source} \sim HP_{target}$$

$$P_{source} = HH^{-1}P_{source}$$

Hence, we can simply design P_{target} as $H^{-1}P_{souce}$. What's more, we can add some translations, which won't hurt the similarity between P_{souce} and P'_{target}, to prevent overlapping or truncation by the image.

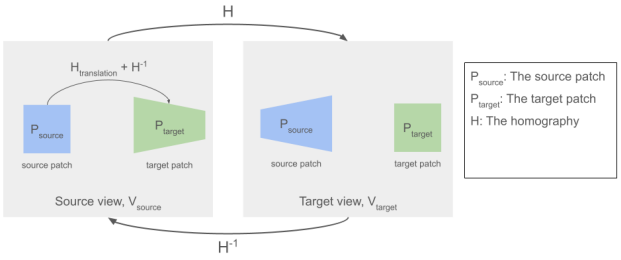

Figure 3: The generation of the masking for two patches

3.4 Dataset

We use HPatches[2] as the dataset to evaluate the performance of the attack. HPatches is composed of two parts. The 59 patches are extracted from the image sequences with the viewpoint changes, and the other 57 patches are extracted from the sequence with the illumination changes. We only take the 59 patches with the viewpoint changes for evaluation.

For each patch, there is one reference image and five compared images with the homography transformation matrix between them. Then, we synthesize the adversarial patches on the images. To fill the mask with the generated patches, we use backward warping with bi-linear interpolation.

In the targeted viewpoint settings, we compute the position and the shape of the mask for each pair of the reference image and the compared image. In the untargeted one, we randomly select a compared image and compute the mask for each scene at the first step. Then, we use the homography matrix provided by the dataset to compute the position of the mask from the previous step for the other viewpoints. Hence, the mask of the same scene will be consistent in the 3D space. The default setting is the targeted viewpoint.

Figure 4: The visual result of the matching of a scene from two viewpoints

3.5 Metrics

Without specification, we select the top-1000 points by k-NN matching.

Source point ratio means the number of the point detected in the source mask of the source view over the number of the point in the source view.

True positive rate means the number of the point that is detected in the source mask of the target view over the number of the the number of the point detected in the source mask of the source view.

False positive rate means the number of the point that is detected in the target mask of the target view over the number of the the number of the point detected in the source mask of the source view.

Repeatability evaluates that the same interest point should be detected for each scene. First, use the ground truth homography to transform the interest points from the source view to the target view. Then, take a pair of points from the source view and the target view that are close enough ($\epsilon = 3$) to the same point.

Homography estimation can be viewed as a downstream task to evaluate the quality of the local feature points. First, match the predicted local feature points point from two views by k-NN. Then, use the RANSAC[4] to predict the transformation matrix. Since directly comparing two homographies is not trivial, we utilize the four corners of the source view. If the position of a corner is closed enough after applying the ground truth homography and the predicted homography, we take it as a correct point.

4 Experimental results

4.1 Targeted and untargeted viewpoint

In this experiment, we evaluate the performance of the three basic patches, chessboard pattern, targeted-class adversarial patch, and untargeted-class adversarial patch, under the targeted viewpoint and the untargeted viewpoint. Table 1 is the result. From the targeted viewpoint, the untargeted-class adversarial patch successfully increases the source point ratio and the false positive rate. However, the chessboard pattern outperforms it in the true positive rate and the homography estimation.

4.2 The size of the mask

In this experiment, we evaluate the performance of the three basic patches, chessboard pattern, targeted-class adversarial patch, and untargeted-class adversarial patch, under three different sizes of the mask. The generated patches are the same as the mask. Table 2 is the result. We can see that

Viewpoint	Patch	SPR. ↑	TP. ↓	FP. ↑	Rep.	Homography estimation ↓		
						$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
benign						0.29	0.51	0.60
targeted	chessboard	0.0605	**0.1560**	0.6371	0.3968	**0.22**	**0.39**	**0.44**
	targeted adv. patch	0.0404	0.1700	0.5157	0.5074	0.30	0.51	0.58
	untargeted adv. patch	**0.1164**	0.1989	**0.7055**	0.5289	0.29	0.48	0.56
untargeted	chessboard	0.0622	**0.1969**	**0.4449**	0.5656	**0.25**	**0.42**	**0.50**
	targeted adv. patch	0.0522	0.2174	0.3313	0.5685	0.32	0.53	0.60
	untargeted adv. patch	**0.1485**	0.3402	0.4394	0.5823	0.29	0.49	0.58

Table 1: The result of the targeted and untargeted view point

the relative performance between different patches remains almost the same. However, it is almost impossible to successfully attack on the homography estimation if the masking size is too small, due to the low source point ratio.

Size of mask	Patch	SPR. ↑	TP. ↓	FP. ↑	Rep.	Homography estimation ↓		
						$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
benign						0.29	0.51	0.60
64	chessboard	0.0234	**0.1396**	0.6032	0.5781	**0.29**	**0.50**	**0.57**
	targeted adv. patch	0.0087	0.2177	0.3064	0.5844	0.29	0.52	0.61
	untargeted adv. patch	**0.0566**	0.1690	**0.6665**	0.5793	0.30	0.53	0.62
128	chessboard	0.0605	**0.1560**	0.6371	0.3968	**0.22**	**0.39**	**0.44**
	targeted adv. patch	0.0404	0.1700	0.5157	0.5074	0.30	0.51	0.58
	untargeted adv. patch	**0.1164**	0.1989	**0.7055**	0.5289	0.29	0.48	0.56
256	chessboard	0.1841	0.2827	0.7086	0.3551	**0.07**	**0.13**	**0.15**
	targeted adv. patch	0.1986	**0.2778**	**0.7307**	0.4886	0.30	0.51	0.59
	untargeted adv. patch	**0.2614**	0.3274	0.7287	0.5221	0.28	0.46	0.52

Table 2: The result of the different size of mask.

4.3 Scale-invariant

In this discussion, we want to test the scale-invariant of the adversarial patch. In other words, will the inconsistent size of the patch and the mask affect the attack? In the meantime, we introduce the augmentation and the initialization from the chessboard into the comparison. Table 3 is the result with 128 as the size of the patch. In the same size scenario, the chess-init patch has the highest performance overall, followed by the chessboard pattern. When the patch size is slightly larger than that of the mask, the relative performance remains almost the same. However, when the patch size is larger, the chessboard outperforms others once again. Besides, the augmentation version of the patch is slightly better than the original untarget class version, but it brings lower performance on the homography estimation.

4.4 Transferability

In this section, we evaluate the transferability to other local feature extractors of our attack. We evaluate our attack on SIFT[8] and SuperPoint[3]. Table 4 is the result. We only focus on two patches, the chessboard pattern and chess-init, based on the performance of the previous results. We can see that these two patterns can successfully attack the SIFT as well. However, the performance is not that well against SuperPoint.

5 Discussions

To the best of our knowledge, we are the first to propose a patch-based adversarial attack against SuperPoint[3] even local feature extraction. We successfully perform the attack on a well-known local feature extraction dataset, HPatches[2] by synthesizing the adversarial patches. Although we

Size of patch	Patch	SPR. ↑	TP. ↓	FP. ↑	Rep.	Homography estimation ↓		
						$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
benign						0.29	0.51	0.60
128	chessboard	0.0605	**0.1560**	0.6371	0.3968	*0.22*	*0.39*	*0.44*
	targeted adv. patch	0.0404	*0.1700*	0.5157	0.5074	0.30	0.51	0.58
	untargeted adv. patch	0.1164	0.1989	0.7055	0.5289	0.29	0.48	0.56
	aug. patch	*0.1791*	0.2464	*0.7360*	0.6502	0.27	0.48	0.56
	chess-init patch	**0.1968**	0.1814	**0.8250**	0.4378	**0.19**	**0.33**	**0.38**
100	chessboard	*0.1612*	**0.1520**	**0.8971**	0.6274	**0.24**	**0.40**	**0.46**
	targeted adv. patch	0.0393	*0.2263*	0.6109	0.5788	0.30	0.53	0.61
	untargeted adv. patch	0.0875	0.2836	0.6642	0.6034	0.31	0.54	0.62
	aug. patch	0.1249	0.3067	0.6634	0.6555	0.33	0.55	0.63
	chess-init patch	**0.1653**	0.2361	*0.7852*	0.6443	*0.29*	*0.48*	*0.55*
150	chessboard	0.0185	**0.1284**	0.0816	0.5813	*0.25*	*0.43*	*0.50*
	targeted adv. patch	0.0331	0.1707	0.5216	0.5708	0.28	0.52	0.59
	untargeted adv. patch	0.1184	0.1916	0.7421	0.5724	0.27	0.49	0.58
	aug. patch	**0.1781**	0.2245	*0.7640*	0.6056	0.26	0.46	0.51
	chess-init patch	*0.1307*	*0.1457*	**0.847**1	0.5341	**0.21**	**0.38**	**0.45**
64	chessboard	**0.0790**	**0.1363**	**0.8783**	0.6562	**0.29**	**0.51**	**0.59**
	targeted adv. patch	0.0081	0.4059	0.1404	0.5925	*0.30*	*0.52*	*0.62*
	untargeted adv. patch	*0.0207*	*0.3076*	*0.4090*	0.5903	*0.30*	0.53	*0.62*
256	chessboard	0.0034	**0.1864**	*0.0706*	0.5874	**0.29**	0.51	0.60
	targeted adv. patch	*0.0079*	0.2812	0.0655	0.5889	0.31	0.51	0.60
	untargeted adv. patch	**0.0096**	*0.2237*	**0.0924**	0.5849	*0.30*	0.51	0.60

Table 3: The result of the scale-invariant at small scale and large scale

Local feature extractor	Patch	SPR. ↑	TP. ↓	FP. ↑	Homography estimation ↓		
					$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
SuperPoint[3]	benign				0.29	0.51	0.60
	chessboard	0.0605	**0.1560**	0.6371	0.22	0.39	0.44
	chess-init patch	**0.1968**	0.1814	**0.8250**	**0.19**	**0.33**	**0.38**
SIFT[8]	benign				0.39	0.59	0.66
	chessboard	**0.0810**	0.4281	0.5888	**0.37**	**0.54**	**0.59**
	chess-init patch	0.0235	**0.1827**	**0.7490**	0.37	0.55	0.59

Table 4: The result of the transferability

have shown some vulnerabilities of the local feature extraction and proposed a simple yet effective method to attack it, there is still a lot more to explore. One of the possible directions is to design stronger patterns, which have a higher scale-invariant, smaller size of the mask. To a certain degree, changing the two patches scenario to one patch only.

What's more, though the feature matching in our evaluation is kNN with RANSAC, there have been many works of the deep-learning-based local feature matching, like SuperGlue[10] and LightGlue[7]. Designing an attack against both deep-learning-based local feature extraction and matching may be challenging and delicate work.

From the perspective of defenses, there have been some works[1] [11] to detect copy-move forgery, which our attack can be somehow classified as. We leave the debate between attacks and defenses of the adversarial attack against the local feature extraction as the future works.

Overall, we hope that this work provides a new perspective on the security of local feature extraction. And we are looking forward to the growth of this topic.

References

[1] Irene Amerini, Lamberto Ballan, Roberto Caldelli, Alberto Del Bimbo, and Giuseppe Serra. A sift-based forensic method for copy–move attack detection and transformation recovery. *IEEE Transactions on Information Forensics and Security*, 6(3):1099–1110, 2011.

[2] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5173–5182, 2017.

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018.

[4] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[6] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[7] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023.

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, nov 2004.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[11] Nor Bakiah Abd. Warif, Ainuddin Wahid Abdul Wahab, Mohd. Yamani Idna Idris, Rosli Salleh, and Fazidah Othman. Sift-symmetry: A robust detection method for copy-move forgery with reflection attack. *Journal of Visual Communication and Image Representation*, 46:219–232, 2017.