# ENLARGING HACKER'S TOOLBOX: DELUDING IMAGE RECOGNITION BY ATTACKING KEYPOINT ORIENTATIONS

*Thanh-Toan Do*[1]*, Ewa Kijak*[1]*, Laurent Amsaleg*[2]*, and Teddy Furon*[3]

[1]Université de Rennes 1, [2]CNRS, [3]INRIA
IRISA, Rennes, France

## ABSTRACT

Content-Based Image Retrieval Systems (CBIRS) used in forensics related contexts require very good image recognition capabilities. Whereas the robustness criterion has been extensively covered by Computer Vision or Multimedia literature, none of these communities explored the security of CBIRS. Recently, preliminary studies have shown real systems can be deluded by applying transformations to images that are very specific to the SIFT local description scheme commonly used for recognition. The work presented in this paper adds one strategy for attacking images, and somehow enlarges the box of tools hackers can use for deluding systems. This paper shows how the orientation of keypoints can be tweaked, which in turn lowers matches since this deeply changes the final SIFT feature vectors. The method learns what visual patch should be applied to change the orientation of keypoints thanks to an SVM-based process. Experiments with a database made of 100,000 real world images confirms the effectiveness of this keypoint-orientation attacking scheme.

***Index Terms***— Forensics, Security, Content-Based Image Retrieval, SIFT, SVM

## 1. INTRODUCTION

CBIRS get increasingly involved in multimedia forensics applications such as the detection of illegal copies of copyrighted material, or the automatic detection of child pornography images. Since the goal is here to scout fraudulent behaviors, the system is likely facing some malevolent forces which will adapt and strike back. Whereas the community usually benchmarks the robustness of CBIRS against generic content transformations (compressions, crops, ...), their security has rarely been addressed. The security of a CBIRS is its ability to resist to some dedicated attacks led by malicious pirates against the specific techniques this system uses. Recently, a handful of papers have warned the community about the poor security levels of CBIRS [2, 1, 3]. These papers describe various strategies endangering the recognition capabilities of systems relying on the well-known SIFT descriptors [5]. Once accurately known the way SIFT works, it is possible to transform images introducing as few visual distortions as possible, while maximizing the impact on the final descriptors used for matching. In turn, the system might fail to recognize an image otherwise always detected, or rank it far in the list of images found to be similar.

In [1], this goal is achieved by using a complex mixtures of keypoint removal and keypoint creation. Removing keypoints reduces the number of matches; creating keypoints produces false positives. Overall, [1] exploits the way the Difference of Gaussian (DoG) value of a keypoint is used in the SIFT extraction process. [1] applies on images carefully crafted visual patches around keypoints to preclude or trigger their detection at feature extraction time.

The work presented in this paper follows a different track. It exploits the way the orientation of keypoints is used in the SIFT extraction process. In Section 2, we review SIFT and then study the influence of the orientation of keypoints on the matching of descriptors. Numerical experiments show that if a sufficient change in orientation of a keypoint is introduced, then the resulting descriptor computed from the attacked keypoint is more difficult to match with the original descriptor. Section 3 proposes a method based on multiple SVMs for learning the best orientation changing patches to be applied on keypoints. Section 4 evaluates the effectiveness of this method against a database of 100,000 images and shows this orientation-based strategy for attacking images described with SIFT can delude systems.

## 2. SIFT: FROM ORIENTATION TO DESCRIPTOR

### 2.1. Overview of SIFT

SIFT computes local features by running a three steps process. First, it detects a keypoint located in $(x, y)$ in the image at scale $\sigma$ if it is a local extremum of the DoG response. In the second step, the main orientation $\theta$ is computed based on gradient directions locally around $(x, y)$. The keypoint is defined as $kp = \{x, y, \sigma, \theta\}$. The third step computes a descriptor on a support region centered on $(x, y)$, whose size depends on scale $\sigma$. A support region is divided into 16 subregions, and a 8-bin quantized histogram of weighted gradient orientation is computed on each subregion, resulting in a 128-dimensional

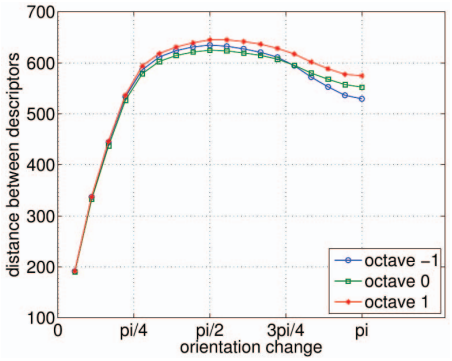


**Fig. 1**. Euclidean distance between descriptors as a function of the distance in radian between the keypoint orientation.

descriptor. It is key to note that gradient orientations are determined *relatively to the keypoint orientation* $\theta$, achieving invariance to rotation.

### 2.2. Changing Orientations Impacts Descriptors

As the value of the descriptor is relative to the keypoint orientation, modifying that orientation will thus change the descriptor. Note that this is different from rotating the whole support region, which has no impact on description.

Figure 1 shows the impact of a forced change of orientation for all the keypoints from octaves $\{-1, 0, 1\}$ of the Lena image. We first described Lena using the open-source VLFeat package [6]. We then patched the code of VLFeat to artificially change the orientation of the keypoints it detects by a multiple of $\pi/18$. We then launched 35 descriptions of Lena, each time increasing the orientation change from $\pi/18$ to $35\pi/18$. The Figure 1 plots the distance in the feature space between the descriptors computed on Lena and the descriptors after the orientation changes. Only the distances for changes between $\pi/18$ and $\pi$ are reported in this figure as it is symmetric beyond that point. Figure 1 shows that the larger distances in feature space are reached when keypoint orientation is changed by $\pi/2$ (hence also $3\pi/2$). In other words, if the new orientation is orthogonal to the original one, then the maximum difference between descriptors is obtained. This applies to other images as well.

## 3. SVM FOR CHANGING ORIENTATIONS

This section describes the method we use to force the orientation of keypoints. In short, we learn what patch $\boldsymbol{\epsilon}$ should be applied to a specific support region to change the orientation by $\pi/2$. This method relies on a collection of SVMs. Each SVM determines the hyperplane separating the keypoints having the orientation $\theta_1$ from the keypoints having the orientation $\theta_2 = \theta_1 + \pi/2$. To facilitate learning, reduce the noise and be more effective, the orientation space is quantized in bins of length $\pi/18$. Therefore, we use 18 SVM classifiers trained for the 18 pairs of orthogonal orientations (e.g., from orientations ranging in $[0, \pi/18]$ to orthogonal orientations ranging in $\pi/2 + [0, \pi/18]$).

To train the SVMs, we first determine all the keypoints and their orientation (using VLFeat) for a set of 1,000 images randomly picked from Flickr. We keep only keypoints belonging to the octaves $\{-1, 0, 1\}$ as their support regions are quite small, facilitating patching them visually while not too severely distorting the images. We then normalize all support regions to be of size $(12 \times 12)$ (this is the average size of lowest octave patches observed on a large set of images) and map all of them from gray-scale to range $[0, 1]$. It is then stored in a vector $\mathbf{r}$ of $L$ components.

The set of keypoints is then divided into classes according to their orientations $\theta$. Let $\mathcal{X}_1 = \{(\mathbf{r}_i, \ell_i)\}_i$ be the training set of normalized support regions $\mathbf{r}_i$ of a given orientation $\theta_1$, forming the class labeled by $\ell_i = +1$. $\mathcal{X}_2 = \{(\mathbf{r}_j, \ell_j)\}_j$ is the training set of normalized support regions $\mathbf{r}_j$ whose orientation is $\theta_2 = \theta_1 + \pi/2$, forming the dual class labeled by $\ell_j = -1$. At training time, the SVM in charge of $\theta_1$ and $\theta_2$ learns the hyperplan parameters $(\mathbf{w}, b)$ separating $\mathcal{X}_1$ and $\mathcal{X}_2$ as solution of:

$$\begin{aligned} \ell_k.(\langle \mathbf{w}, \Phi(\mathbf{r}_k)\rangle + b) \geq 1 \qquad & \forall \mathbf{r}_k \in \{\mathcal{X}_1, \mathcal{X}_2\}, \\ \text{with } \langle \mathbf{w}, \Phi(\mathbf{x})\rangle = \; & \textstyle\sum_{k:\alpha_k>0} \alpha_k \ell_k K(\mathbf{x}, \mathbf{r}_k), \end{aligned}$$

where $\Phi$ maps $\mathbf{x}$ to an higher dimensional space, $\alpha_k$ are the Lagrange multipliers, and $K$ is the Radial Basis kernel function (RBF):

$$K(\mathbf{x}, \mathbf{r}_k) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{r}_k)\rangle = \exp\left(-\frac{\|\mathbf{x} - \mathbf{r}_k\|^2}{2\sigma^2}\right). \quad (1)$$

Once trained, the SVM is used to determine the patch $\boldsymbol{\epsilon}$ of minimum norm to be added to a region $\mathbf{r} \in \mathcal{X}_1$, such that $\mathbf{r} + \boldsymbol{\epsilon} \in \mathcal{X}_2$. This asks to solve the following optimization:

$$\min \frac{1}{2}\|\boldsymbol{\epsilon}\|^2 \quad (2)$$

$$\text{s.t.} \quad \sum_{k:\alpha_k>0} \alpha_k \ell_k K(\mathbf{r} + \boldsymbol{\epsilon}, \mathbf{r}_k) + b = \ell_2 \Delta d, \quad (3)$$

$$\text{and} \quad 0 \leq r_i + \epsilon_i \leq 1, \; \forall i \in \{1, \ldots, L\} \quad (4)$$

where $\Delta d > 0$ is the distance from $\mathbf{r} + \boldsymbol{\epsilon}$ to the hyperplane $(\mathbf{w}, d)$. Eq. (4) ensures that the modified region remains in the range $[0, 1]$. Define scalars $a_k = \alpha_k \ell_k K(\mathbf{r}, \mathbf{r}_k)$, and vectors $\mathbf{c}_k = 2(\mathbf{r} - \mathbf{r}_k)$. Then, Eq. (3) can be rewritten as:

$$\sum_{k:\alpha_k>0} a_k \exp\left(-\frac{\mathbf{c}_k^\top \boldsymbol{\epsilon} + \|\boldsymbol{\epsilon}\|^2}{2\sigma^2}\right) = \ell_2 \Delta d - b. \quad (5)$$

This minimization problem under constraints is solved using an interior-point method, resulting in the desired $\boldsymbol{\epsilon}$ to be applied. Once $\boldsymbol{\epsilon}$ is known for a particular $\mathbf{r}$, we reshape to $12 \times 12$, interpolate to fit the corresponding support region, and rescale it, to finally add this patch to the pixels.

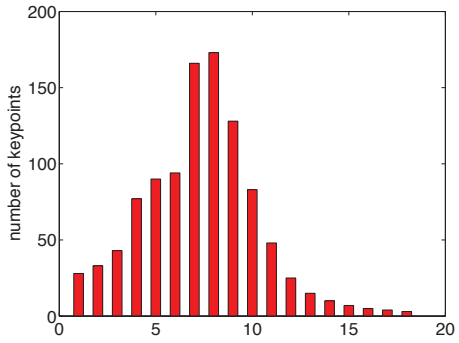


**Fig. 2**. Number of keypoints per orientation change bin after support region modification.

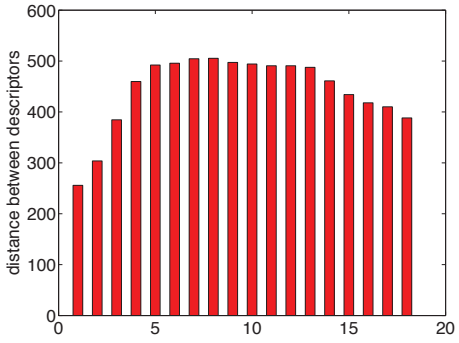


**Fig. 3**. Distance between the original and modified descriptors per orientation change bin.

## 4. EVALUATION

In this section, we first evaluate the effectiveness of the above method for changing the orientation of keypoints. We show that while the orientation of many keypoints indeed change, some remain un-impacted. We evaluate the method at the image level and show new keypoints appear as side effects of visual distortions. We finally benchmark the effectiveness of the method when querying a database of 100,000 random images with orientation-attacked quasi-copies.

The 100,000 images used have been randomly downloaded from Flickr and are very diverse in contents. All images have been resized to 512 pixels on their longer edge. This collection yields 103,454,566 SIFT-VLFeat descriptors. The SVMs were trained using 1,000 random images from that set, as described earlier. Note that this amounts to about 1,026,000 samples, and the number of samples per orientation class ranges between 19,567 and 45,060. Note also that we set $\Delta d = 2$ in our experiments.

### 4.1. Ability to Change Orientations

We applied the method to the keypoints of the 1,000 images. To check whether or not orientations changed, we observed for all keypoints the angle between each original and attacked keypoint, expecting a change of $\Delta\theta = \pi/2$. However, this is not always verified. Figure 2 counts the number of keypoints as a function of the observed orientation change $\Delta\theta$. Each bin on x-axis covers a range of $\pi/18$ from 0 to $\pi$: the first bin corresponds to keypoints with $\Delta\theta \in [0, \pi/18[$, ... It appears that for most of the keypoints, the orientation is changing by $6\pi/18$ to $8\pi/18$ ($7^{th}$ and $8^{th}$ bins). The value of $\Delta d$ drives this phenomenon: a larger value for $\Delta d$ increases the number of $\pi/2$ changes but in turn causes more severe and visible distortions in the patches.

While Figure 1 suggested enforcing $\Delta\theta = \pi/2$ would be best, Figure 3 shows that, in practice, a value for $\Delta\theta$ ranging from $4\pi/18$ to $13\pi/18$ pushes far away the attacked descriptors in the feature space. Figure 3 shows the average of the euclidean distances between the original and attacked descriptors, as a function of $\Delta\theta$. The observed distances are fairly constant between $4\pi/18$ and $13\pi/18$. It is therefore not necessary to increase $\Delta d$. When $\Delta d = 2$ and when combining Fig. 2 and 3, then about 79% of the orientations changed, moving as far as possible descriptors in the feature space, and making matches potentially problematic.

The average PSNR between the original and attacked *patches* is 21.64dB. It is possible to preserve the PSNR by running a small variant of the method. Instead of applying the patch to the whole support region, only it central region is added. The size of the latter is proportional to the size of the support region, e.g. $11 \times 11$ for scale $\sigma = 0$. It is however quite effective due to the weighting scheme used when determining the orientation, the central area of the support regions having more influence. Reproducing all experiments with this variant gives an average PSNR of 23.84dB. It does, however, change the effectiveness of the method as fewer keypoints have their orientation changed: modifying the central part of their support region being not sufficient. Figure 4 counts the number of keypoint as a function of $\Delta\theta$ observed with this variant. It clearly shows many orientations could not be changed (see the left-most bin); most of the keypoints that changed orientation have a $\Delta\theta \in [4\pi/18, 10\pi/18]$. The distances between descriptors are identical to the ones observed on Fig. 3. Because it gives good results while preserving the PSNR, this variant is the method used in the sequel.

### 4.2. Impact at Image-Level

To get an acceptable visual distortion for the attacked image, the variant modifying the center of support regions is used and applied only if the PSNR between the original and patched support regions is bigger than a given threshold $t_{PSNR}$. We apply the method to 1,000 images for 3 different $t_{PSNR}$ values: 15.3, 16.3, and 17dB. The average $PSNR$ computed over the 1,000 images are respectively 28.39, 29.24, and 29.93dB, showing the PSNR increases with $t_{PSNR}$.

Having a closer look on how the keypoints are modified by the orientation attack shows that keypoints can be divided into three classes: (i) either the keypoint is unchanged (same

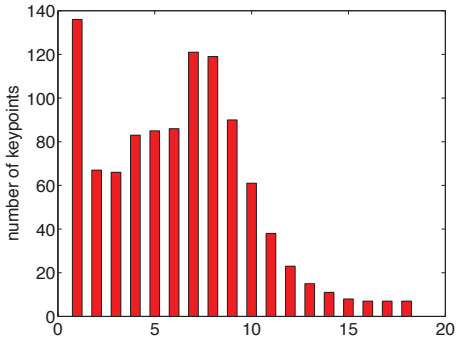


**Fig. 4**. Number of keypoints per orientation change bin when applying a centered cropped patch.

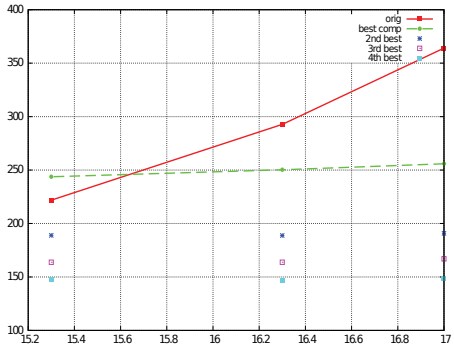


**Fig. 5**. Average scores over 1,000 queries for $t_{PSNR}$ equals 15.3, 16.3, or 17.

location, same scale, same orientation), or (ii) its orientation has changed (same location, same scale, but significantly different orientation), or (iii) a new keypoint is has been created as side effect of distortions introduced by the attack.

We evaluate how many keypoints fall into each class as follows. Let $kp_o = \{x_o, y_o, \sigma_o, \theta_o\}$ be an original keypoint, $(x, y, \sigma, \theta)$ a keypoint in attacked image, and *d(.,.)* the euclidean distance. A keypoint falls into the class (i) if there is a $kp_o$ such that $d\left((x, y), (x_o, y_o)\right) \leq 5$ and $0.7 \leq \sigma/\sigma_o \leq 1.3$ and $|\theta - \theta_o| \leq \pi/18$. These values have been determined because any keypoint in that class remains pretty close to its original keypoint in the feature space (at a distance lower than 200, see Fig. 3), allowing easy matching. A keypoint falls into the class (ii) if there is a $kp_o$ such that $d\left((x, y), (x_o, y_o)\right) \leq 5$ and $0.7 \leq \sigma/\sigma_o \leq 1.3$ and $|\theta - \theta_o| \geq \pi/18$. The remaining keypoints fall into the class (iii). They can be seen as new keypoints as they are far in position or scale with respect to the original keypoints. Overall, when applying the method to 1,000 images with $t_{PSNR} = 17$, then about 58% of the keypoints have their orientation changed (they fall into class (ii)), 28% fall into class (i) and 27% fall into class (iii).

### 4.3. Impact on Large-Scale Recognition

To run an experiment involving large-scale recognition, we indexed the 103,454,566 SIFT-VLFeat descriptors with the NV-Tree high-dimensional indexing scheme [4]. The NV-Tree runs approximate $k$-NN queries and has been specifically designed to index large collections of local descriptors. The same 1,000 images are used as queries and we ran the proposed orientation attack on them resulting in quasi-copies. The variant modifying the center of support regions is used and controlled by the $t_{PSNR}$ threshold. Each query probes the system which returns the top 100 images with associated scores. We then compute the average score of the original image (the one used to forge the quasi-copy that the system *should* identify). Figure 5 shows average score of original image (red line) and the four top matching images for some different $t_{PSNR}$ values. From right to left, the gap between original image and best competitor scores decreases, as the strength of the attack increases. The attack succeeds for $t_{PSNR} = 15.3$. Even if the attacked image is not completely concealed, the original image has not the best score anymore, and gets hidden behind another image that better matches.

## 5. CONCLUSIONS

A new angle of attack of CBIRS based on SIFT descriptors, focusing on the influence of the orientation disturbance on the recognition of an image, is studied. The orientation shift in descriptor computation is accomplished by introducing locally non-affine modifications, through the addition of patches that are learned by an SVMs process. The effectiveness of the method is evaluated on a substantial number of images. The results show that applying this single attack can lower enough the score of the original image so that it is no longer returned at first position by the system. Clearly, to be truly effective, this strategy must be combined with other attacks. In future work, the geometric verification step included in CBIRS as post-filtering should be considered also.

## 6. REFERENCES

[1] T.-T. Do, E. Kijak, T. Furon, and L. Amsaleg. Deluding Image Recognition in SIFT-based CBIR Systems. In *ACM Multimedia in Forensics, Security and Intelligence*, 2010.

[2] T.-T. Do, E. Kijak, T. Furon, and L. Amsaleg. Understanding the security and robustness of sift. In *ACM Multimedia*, 2010.

[3] C.-Y. Hsu, C.-S. Lu, and S.-C. Pei. Secure and robust sift. In *ACM Multimedia*, 2009.

[4] H. Lejsek, F. H. Amundsson, B. T. Jonsson, and L. Amsaleg. NV-Tree: An efficient disk-based index for approximate search in very large high-dimensional collections. *IEEE TPAMI*, 31(5), 2009.

[5] D. Lowe. Distinctive image features from scale invariant keypoints. *IJCV*, 60(2), 2004.

[6] A. Vedaldi and B. Fulkerson. VLFeat - An open and portable library of computer vision algorithms. In *ACM Multimedia*, 2010.