Counter-Forensics of SIFT algorithm by means of keypoint classification: a copy-move scenario

I. Amerini[2], M. Barni[1], R. Caldelli[2], A. Costanzo[1]

[1]Department of Information Engineering, University of Siena, Italy
[2]Media Integration and Communication Center, University of Florence, Italy

October 2^{nd}, 2012

Outline

Introduction

Introduction: the problem I

- SIFT (Scale Invariant Feature Transform) features have become extremely popular in a wide variety of fields
- In some of these applications the security of SIFT is an important matter
  - image search engines (Content Based Image Retrieval)
  - copyright infringement detectors
  - image forensics detectors

It has become crucial to understand to what extent one can entrust SIFT with such delicate tasks

Introduction: the problem II

- This endeavor is carried out by devising methods (i.e. attacks) to force SIFT to fail its task
- Not a *divertissement*, exposing weaknesses is becoming as important as exploiting strengths
  - see the growing interest on counter–forensic
- *So, what has been found so far? Is SIFT secure?*
- Some weaknesses have been found. However, attacks are definitely not simple [1, 2, 3, 4, 5]

Introduction: our contribution I

- Literature so far devised a fair number of attacks assuming equal properties for all features (i.e. keypoints)
- On the contrary, we suppose they are not equal
  - some features react better than others to certain attacks
- Our contribution is twofold

Introduction: our contribution I

- Literature so far devised a fair number of attacks assuming equal properties for all features (i.e. keypoints)
- On the contrary, we suppose they are not equal
  - some features react better than others to certain attacks
- Our contribution is twofold

> We propose a **classification criterion** based on first order statistics (histogram)

Introduction: our contribution I

- Literature so far devised a fair number of attacks assuming equal properties for all features (i.e. keypoints)
- On the contrary, we suppose they are not equal
  - some features react better than others to certain attacks
- Our contribution is twofold

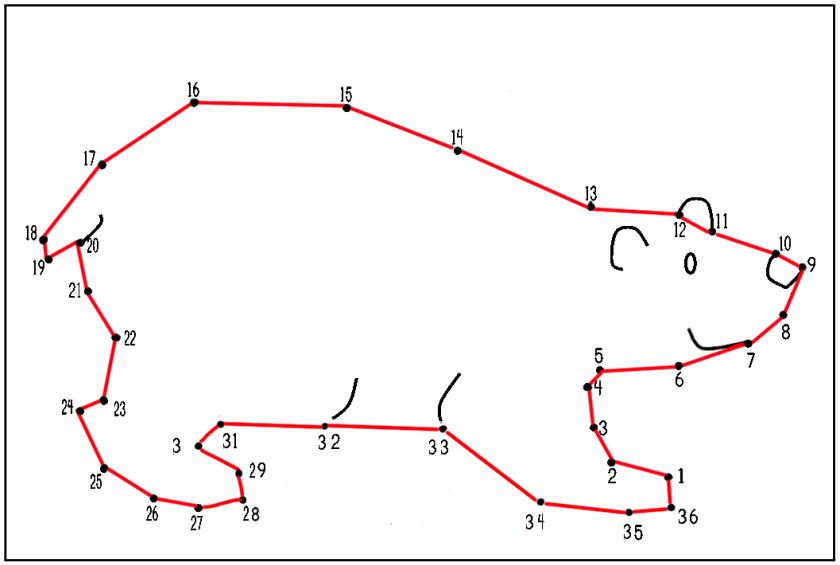
> We propose a **classification criterion** based on first order statistics (histogram)

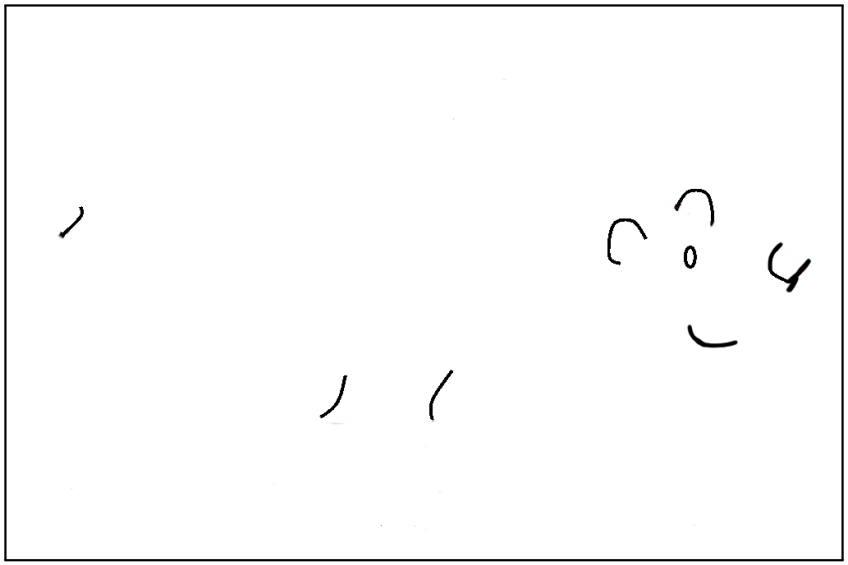
> We build an **iterative algorithm that removes SIFT features with class-tailored attack**

Introduction: our contribution II

- Two common assumptions in literature
  - gray scale images (SIFT ignores colors)
  - first scale keypoints
    - majority
    - most stable thus difficult to remove
    - significant yet not excessive amount
- Achievements
  - outperforming existing attacks applied blindly to all keypoints
  - robust against different SIFT implementations
  - successfully disabling a SIFT-based copy-move forgery detector

Introduction: our contribution III





SIFT

SIFT [6]: what

SIFT OBJECTIVE

A set of features invariant to scaling, rotation, cropping, change of point of view, change of lighting

- A feature $\mathbf{x} = (x, y, \sigma, \Theta)$ is called keypoint
  - (x, y) are the *spatial coordinates*
  - σ is the *scale* (related to Gaussian smoothing)
  - Θ is the *orientation*
- Given two similar images, features are matched by comparing an unique fingerprint (*descriptor*) assigned to each of them
  - no choice on SIFT steps up to the descriptor
  - complete freedom for the matching step

SIFT: Step 1 – building scale space

- Steps 1–2 achieve scale and affine transform invariance

Image I is repeatedly convolved with Gaussians to obtain the smoothed image L; Adjacent smoothed images are subtracted to obtain DoG (Difference of Gaussians)

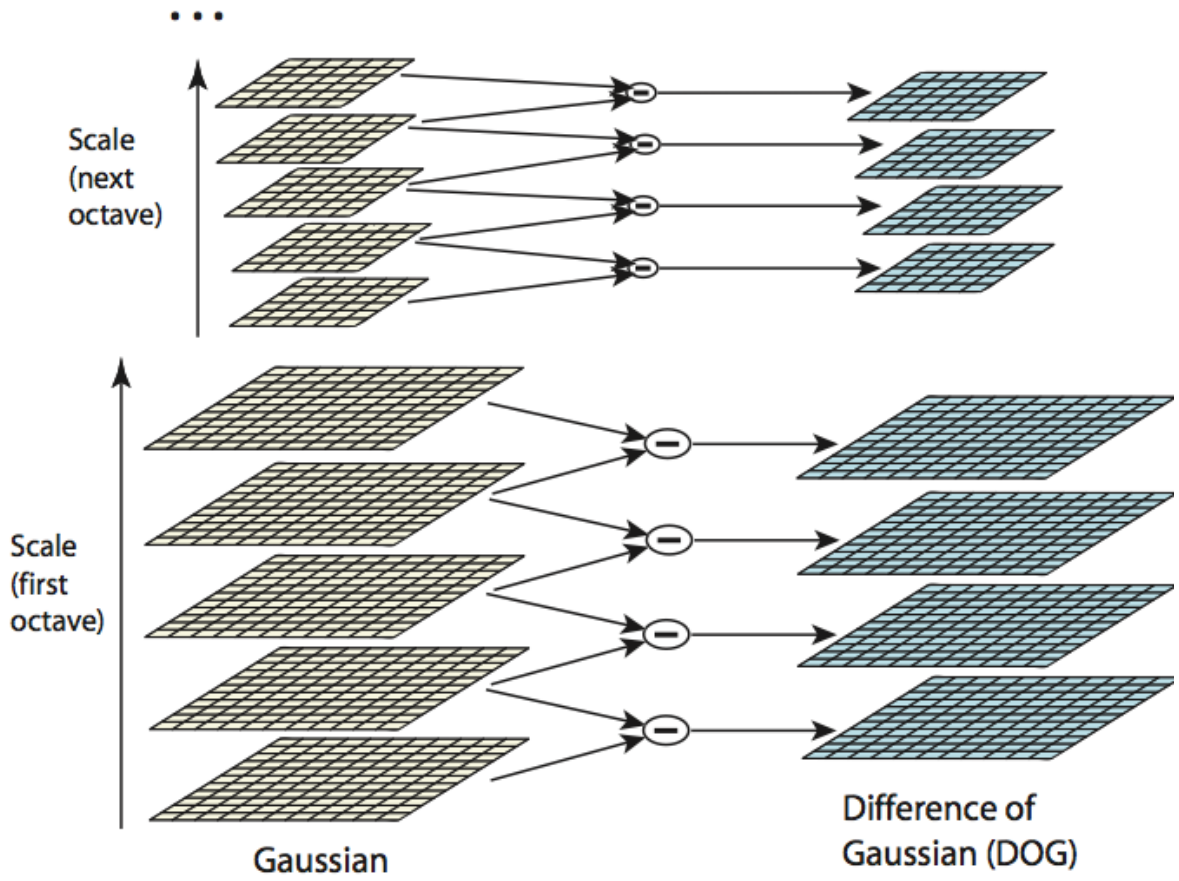
$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y)$$

$$D(x, y, \sigma) = L(x, y, k_1\sigma) - L(x, y, k_2\sigma)$$

- $G(x, y, \sigma)$ is down-sampled by a factor 2, then the process repeated

SIFT: Step 1 – building scale space

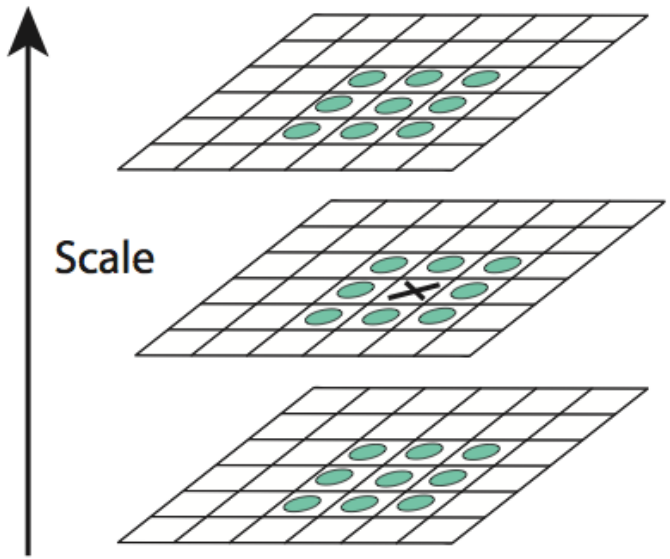
- Steps 1–2 achieve scale and affine transform invariance



SIFT: Step 2 – detecting extrema of DoG

- Steps 1–2 achieve scale and affine transform invariance

Extrema are detected by comparing a pixel (marked with X) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with circles)



- These are the candidate keypoints. Their positions can be refined by sub-pixel approximation. They are *too many*

SIFT: Step 3 – reducing the population

- **Step 3.1**: eliminating low contrast keypoints

  If the magnitude of the DoG at the current keypoint is less than a certain value, the keypoint is rejected

- **Step 3.2**: eliminating edge keypoints

  Calculate 2 perpendicular gradients at the keypoint:

  - Both gradients small: flat region
  - One big gradient: edge
  - Two big gradients ($\geq$ *Threshold*): corner

  Corners make great keypoints, rest is rejected

SIFT: Step 4 – achieving rotation invariance

- **Step 4.1** Compute gradient magnitudes and orientations

Let $L(u, v, \sigma)$ be the blurred image; for each pixel (u, v) in the neighborhood of keypoint:

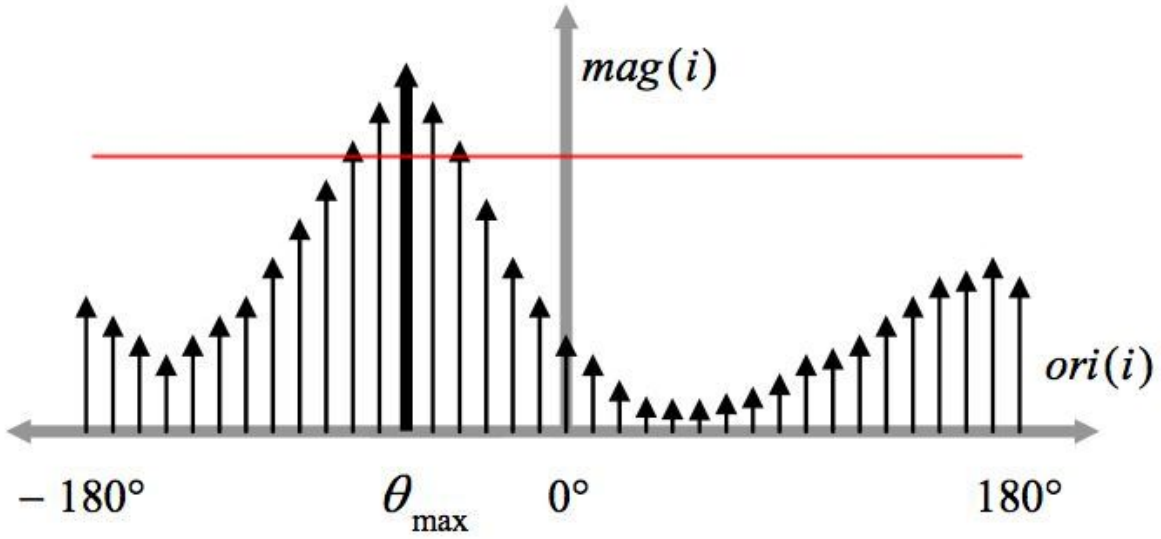
$$m(u,v) = \sqrt{(L(u+1,v) - L(u-1,v))^2 + (L(u,v+1) - L(u,v-1))^2}$$

$$\theta(u,v) = tan^{-1}\Big((L(u,v+1) - L(u,v-1))/(L(u+1,v) - L(u-1,v))\Big)$$

SIFT: Step 4 – achieving rotation invariance

- **Step 4.2**: Compute orientation histograms

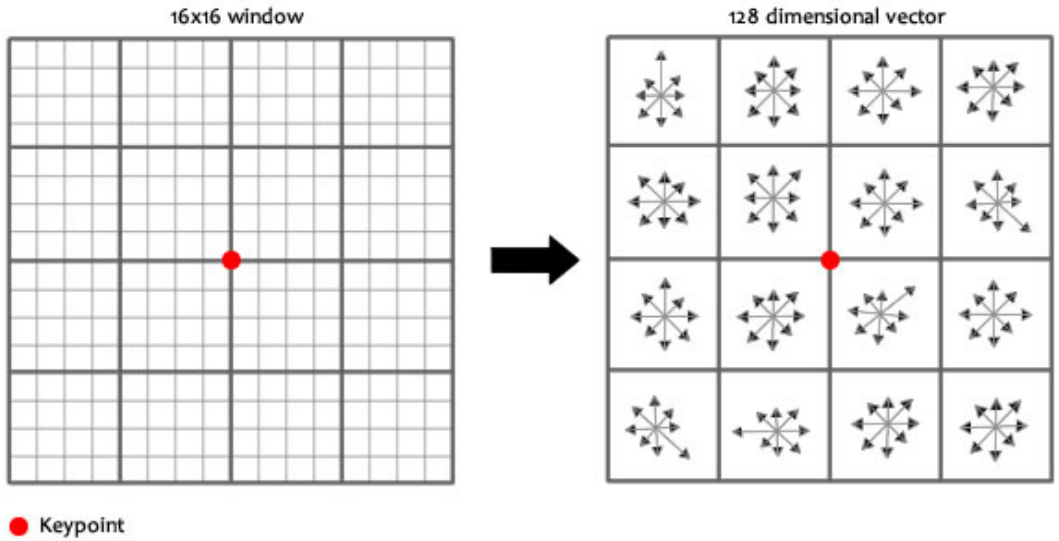
Histogram (36 bins) of $\theta(u, v)$ weighted with $m(u, v)$. The *peak* is the keypoint's dominant orientation Θ. Orientations $\hat{\theta}_i \geq 0.80 \cdot \Theta$ generate new keypoints $\hat{\mathbf{x}}_\mathbf{i} = (x, y, \sigma, \hat{\theta}_i)$



SIFT: Step 5 – computing descriptor

- Achieving uniqueness, lighting, point of view invariance

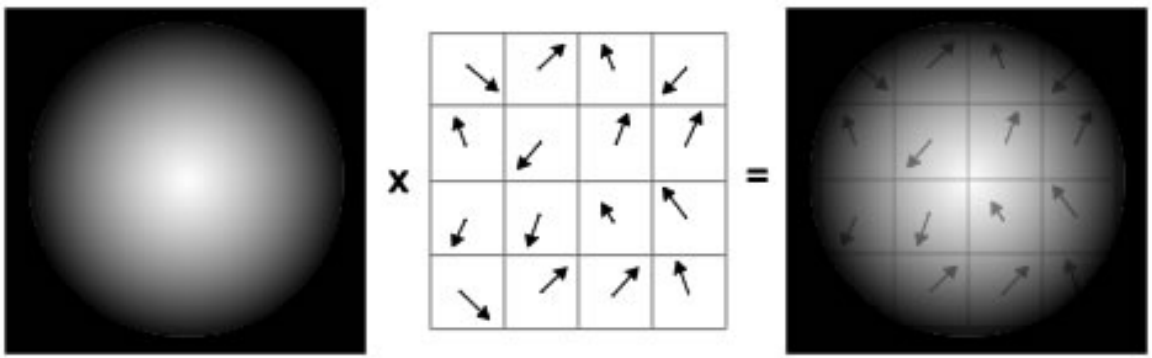
  - A grid centered on keypoint, with orientation from previous step and spacing depending on scale
  - Compute local gradients directions and gather them in a 8–bins histogram



SIFT: Step 5 – computing descriptor

- Achieving uniqueness, lighting, point of view invariance

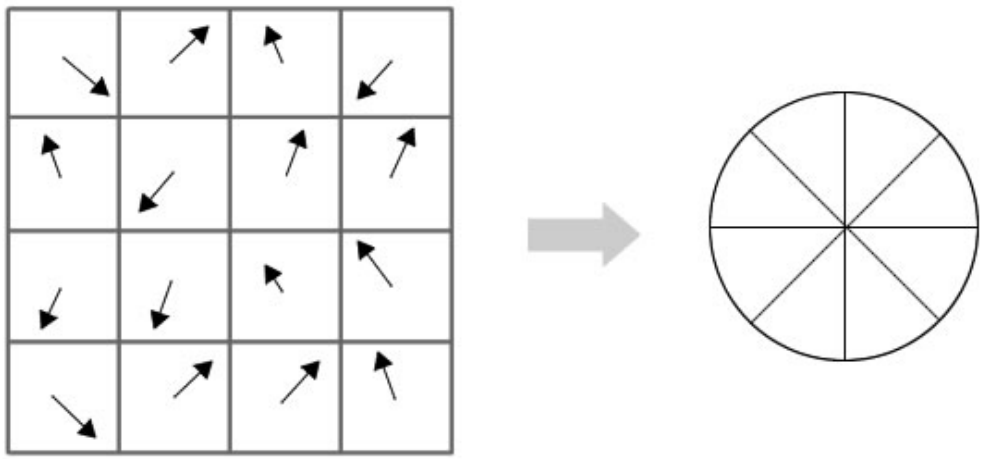
  - A grid centered on keypoint, with orientation from previous step and spacing depending on scale
  - Compute local gradients directions and gather them in a 8–bins histogram



SIFT: Step 5 – computing descriptor

- Achieving uniqueness, lighting, point of view invariance

  - A grid centered on keypoint, with orientation from previous step and spacing depending on scale
  - Compute local gradients directions and gather them in a 8–bins histogram



SIFT: Step 6 – matching descriptors

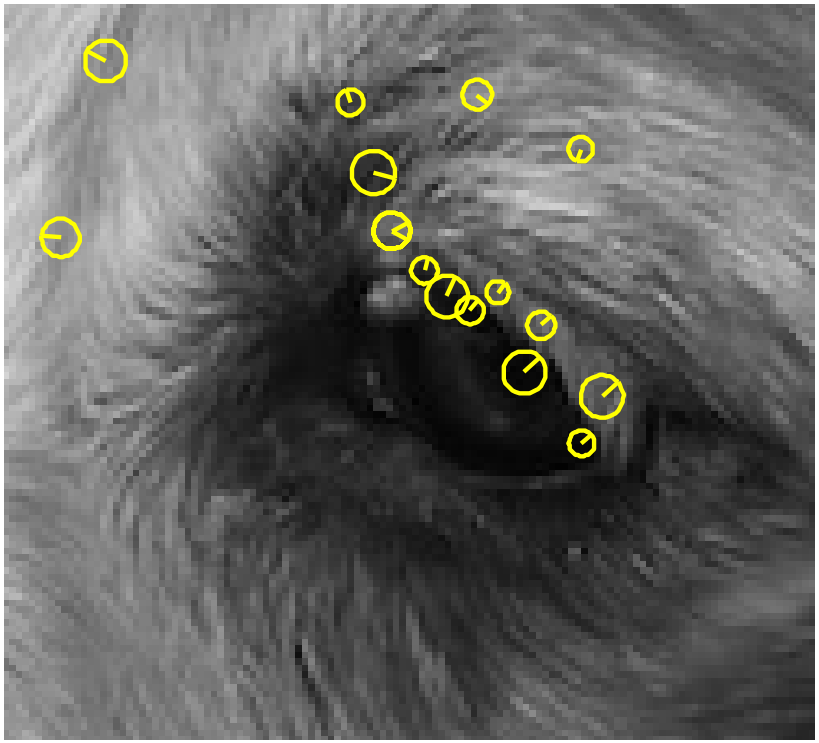
- Descriptors of features coming from two similar images can be mutually matched
  - Minimize the Euclidean distance between the descriptors represented as 128-dimensional vectors
  - Accept ratio between the distances to the nearest and the next nearest points ≤ 0.8
- Several improvements have been proposed
  - increasing accuracy, speed

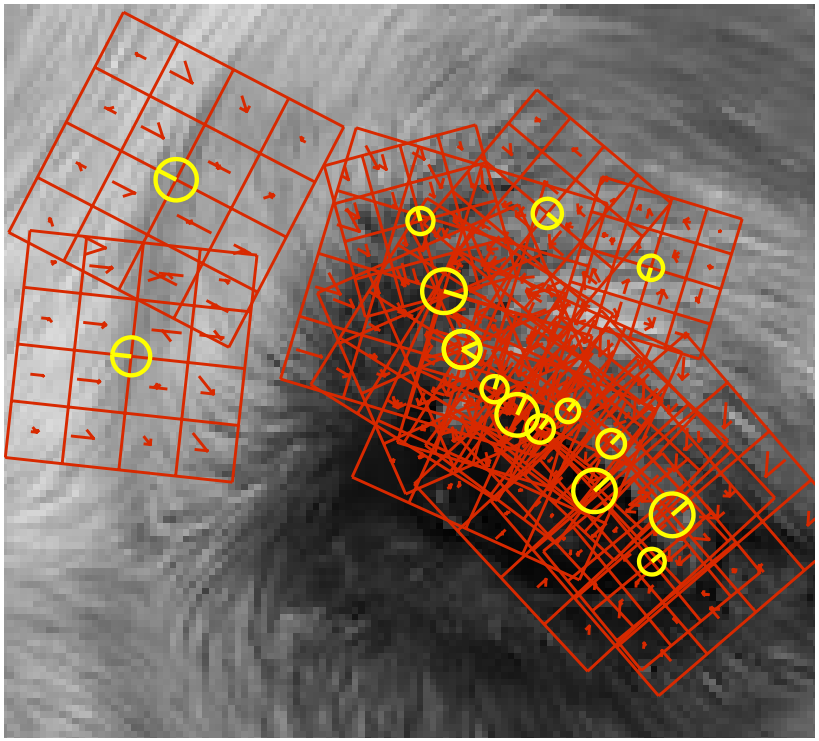
SIFT: example of detection I

- Spatial locations of SIFT keypoints

SIFT: example of detection II

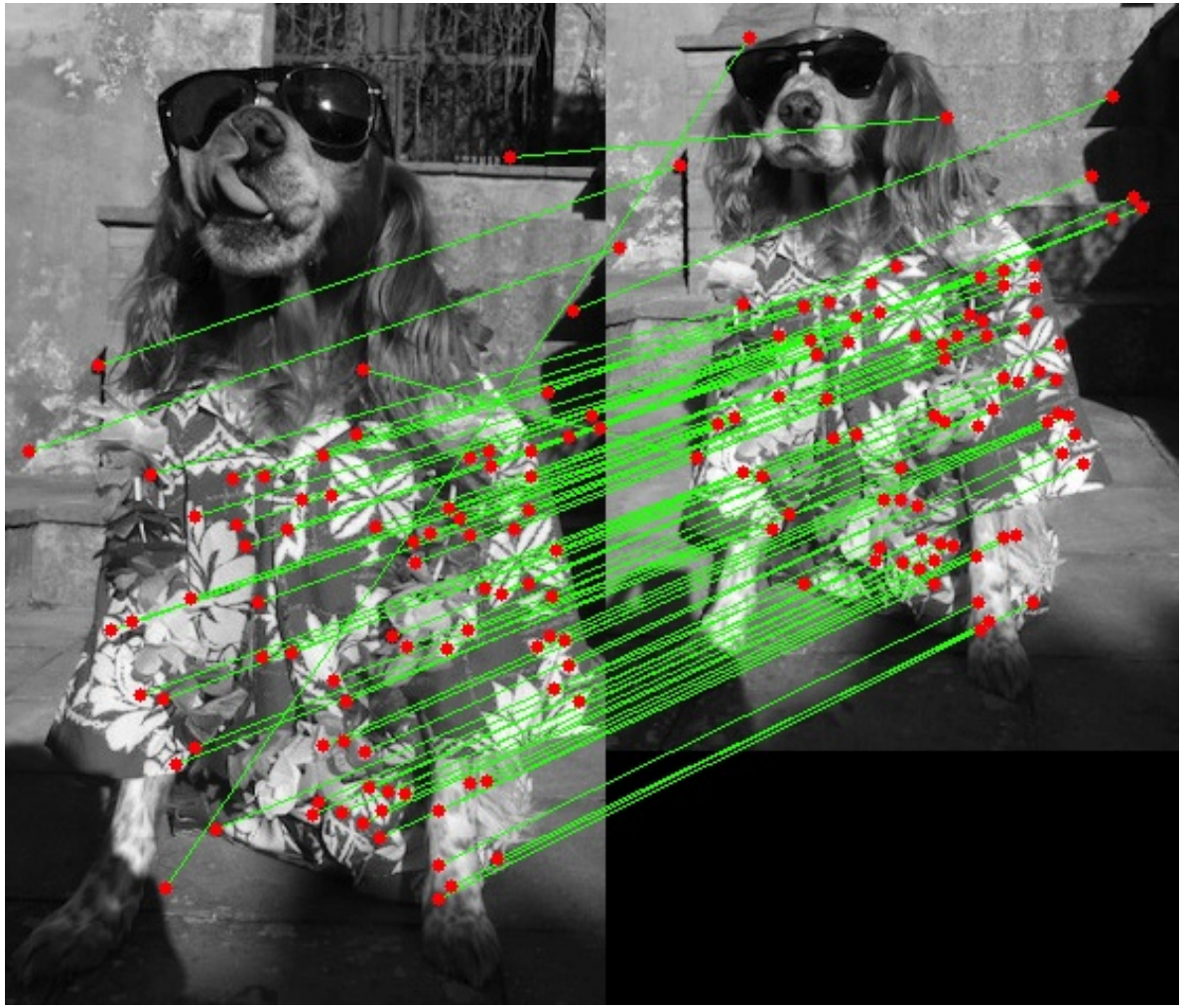
- Spatial locations of SIFT keypoints with dominant orientations (left) and descriptors (right) – detail





SIFT: example of matching

SIFT: example of matching



SIFT-based copy-move detection

Copy-move: what is that? [7]

- Common forgery whereby a portion of an image is copied and pasted once (or more) elsewhere into the same image

Copy-move: what is that? [7]

- Common forgery whereby a portion of an image is copied and pasted once (or more) elsewhere into the same image

Copy-move: SIFT-based detection I

- Older techniques based on pairing of duplicate blocks
  - hardly robust against rotation, scaling and shearing operations
- Problem solved by techniques based on SIFT matching

Copy-move: SIFT-based detection I

- Older techniques based on pairing of duplicate blocks
  - hardly robust against rotation, scaling and shearing operations
- Problem solved by techniques based on SIFT matching

downscale

upscale

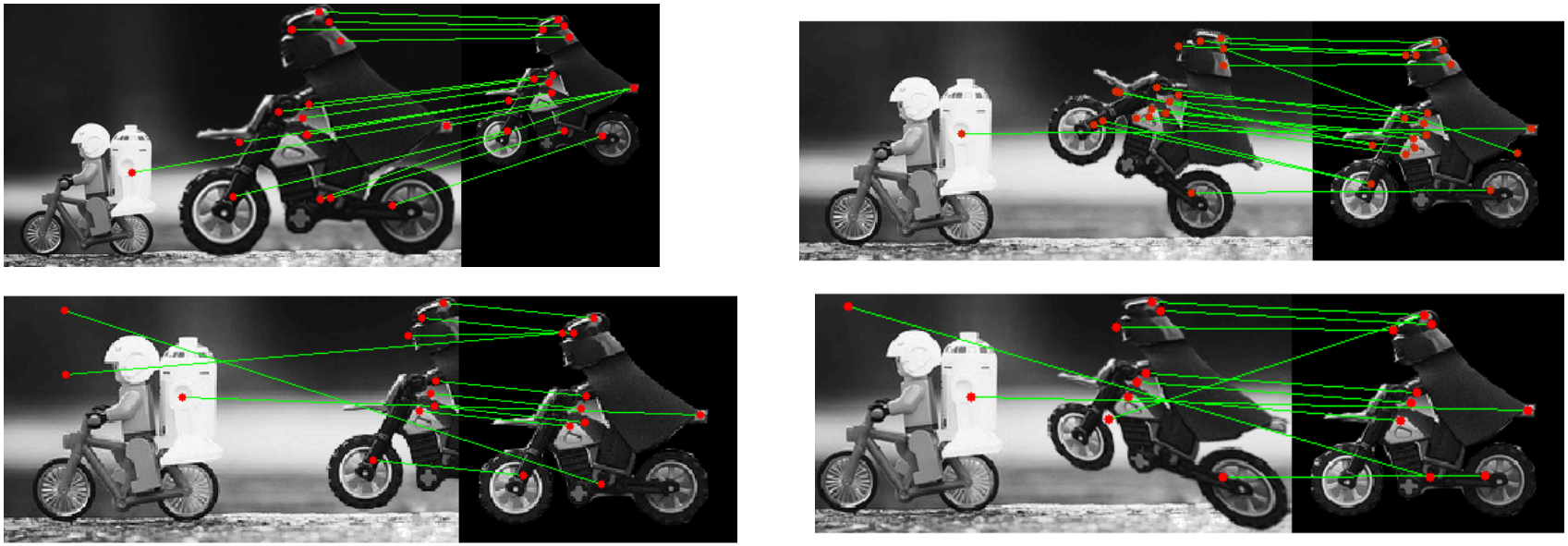
rotate

crop

distort

original

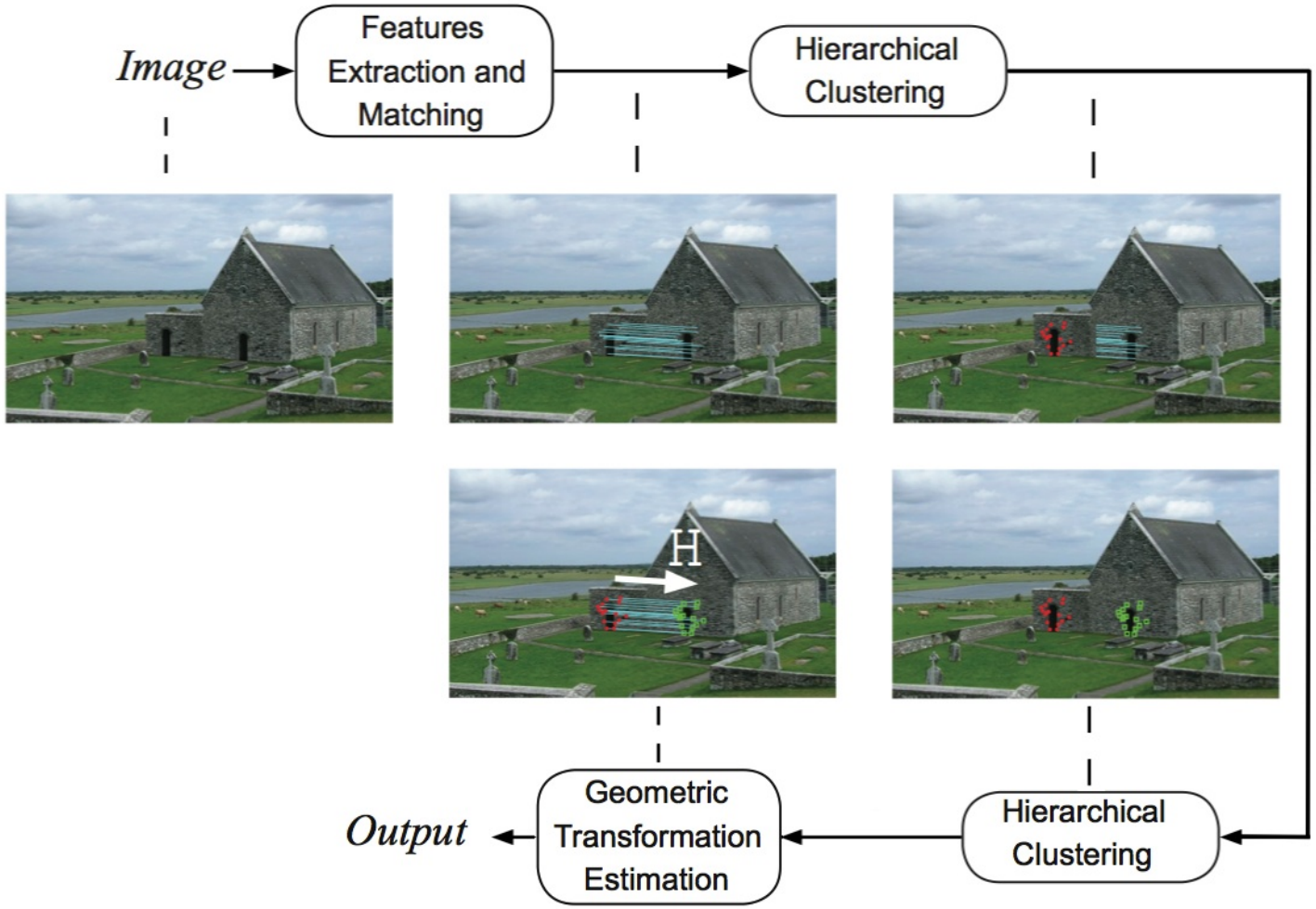
Copy-move: SIFT-based detection II



- ☺ Region tampered if number of matches $\geq$ *Threshold*
- ☹ Attacks to detector aim to reduce matches $<$ *Threshold*
- ☺ Few matches to find copy-move (e.g. $T \geq 3$)
- ☹ No need to delete all keypoints / matches

Copy-move: the chosen detector [8] I

- I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, *A SIFT-based forensic method for copy move attack detection and transformation recovery*, Information Forensics and Security, IEEE Transactions on, 6(3):1099–1110, Sept. 2011



Copy-move: the chosen detector II

Keypoints are extracted from the input image

Copy-move: the chosen detector II

Keypoints are extracted from the input image

For each keypoint the Euclidean distance between its descriptor and those of the rest of keypoints is computed. If *distance* $\geq$ *Threshold* two keypoints are matched

Copy-move: the chosen detector II

Keypoints are extracted from the input image

For each keypoint the Euclidean distance between its descriptor and those of the rest of keypoints is computed. If distance $\geq$ *Threshold* two keypoints are matched

Draft idea of cloned areas, that are refined by means of *agglomerative hierarchical clustering*

Copy-move: the chosen detector II

Keypoints are extracted from the input image

For each keypoint the Euclidean distance between its descriptor and those of the rest of keypoints is computed. If distance $\geq$ *Threshold* two keypoints are matched

Draft idea of cloned areas, that are refined by means of *agglomerative hierarchical clustering*

If two (or more) clusters with at least 4 pairs of matched points linking them, then the image is tampered

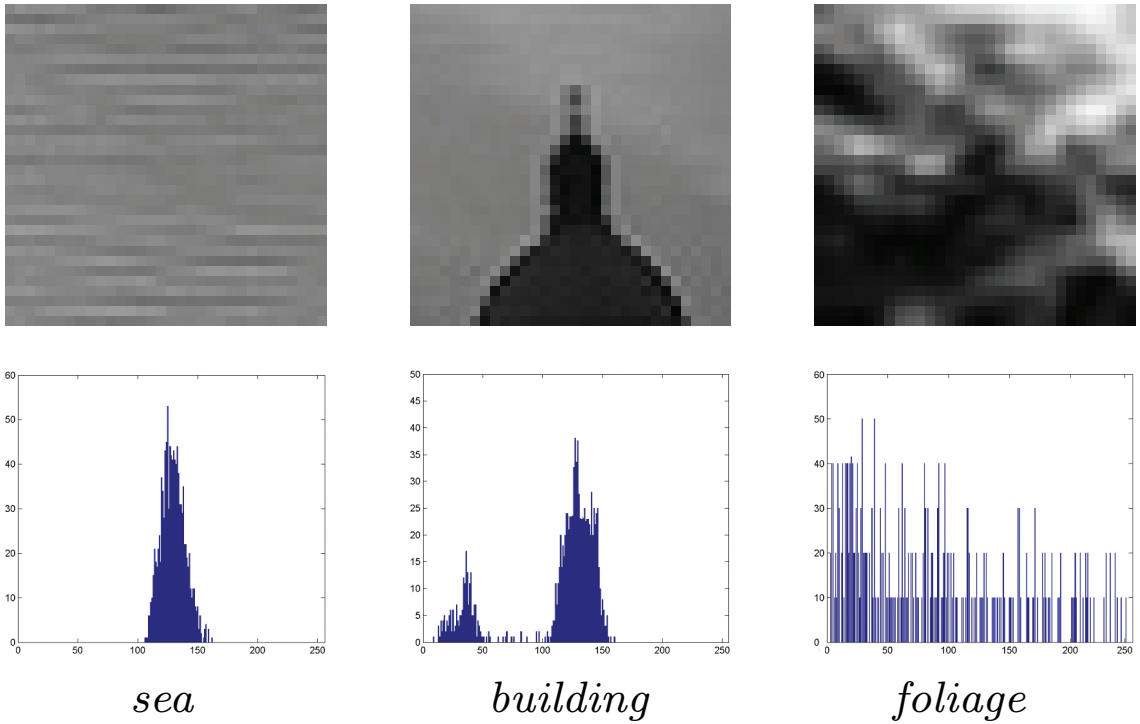
Copy-move: the chosen detector III

Classification of SIFT keypoints

Classification: rationale I

- Distinction based on the gray scale histogram of a small region surrounding the keypoint
- More specifically, on the number of modes M
  - they provide valuable informations about the local content
- Three different classes, three different contents
  - unimodal ($M = 1$): uniform flat regions with low variance (e.g. sky, sea, grass)
  - bimodal ($M = 2$): edges and geometric shapes (e.g. buildings)
  - multimodal ($M > 2$): regions with high variance resembling some sort of noise (e.g. foliage)

Classification: rationale II



Classification: algorithm I [9]

- Originally designed for image segmentation based on histogram thresholding
- Image histogram H modeled as a mixture of Gaussians

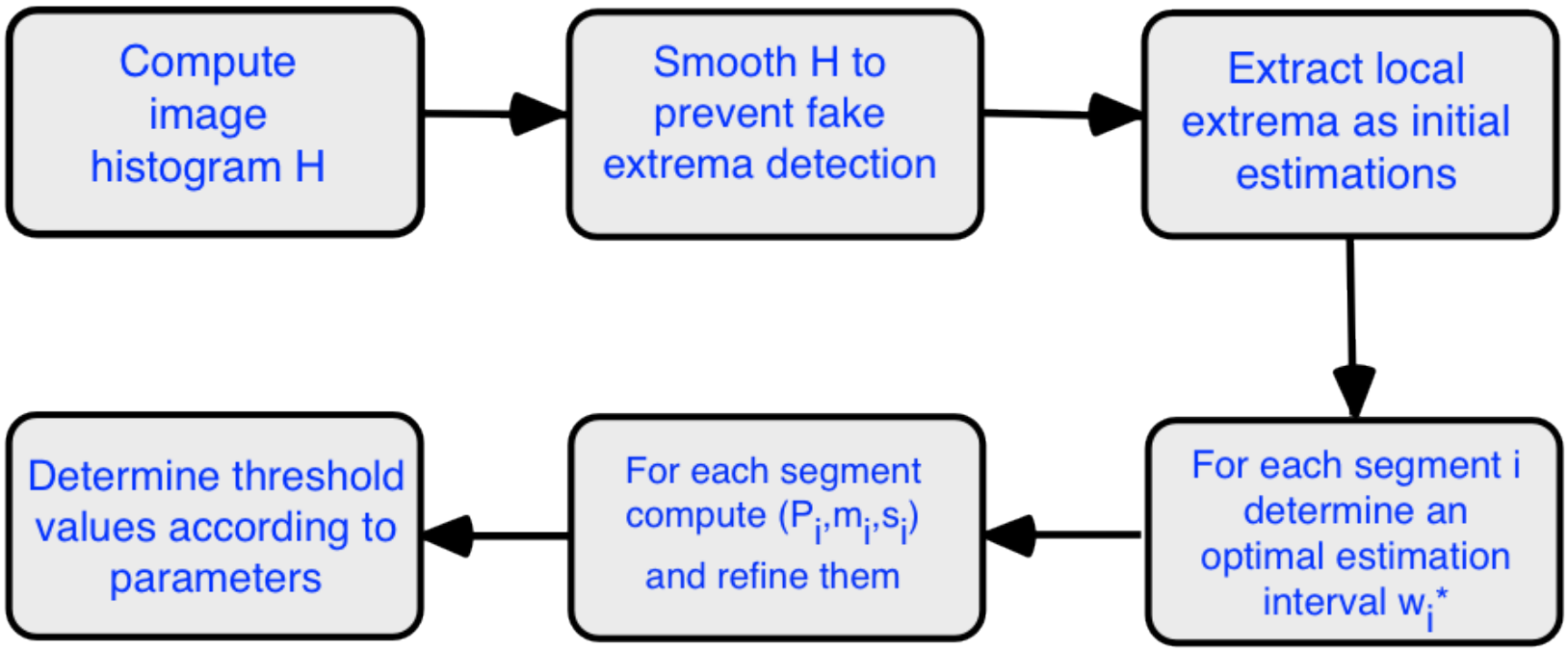
$$f(k) = \sum_{i=1}^{n+1} \frac{P_i}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{k-m_i}{\sigma_i}\right)^2}$$

  - $n+1$ histogram segments
  - $(P_i,\ m_i,\ \sigma_i^2)$ weight, mean and variance of i-th Gaussian

- The classifier estimates the model parameters and minimizes $|f - H|$

Classification: algorithm II

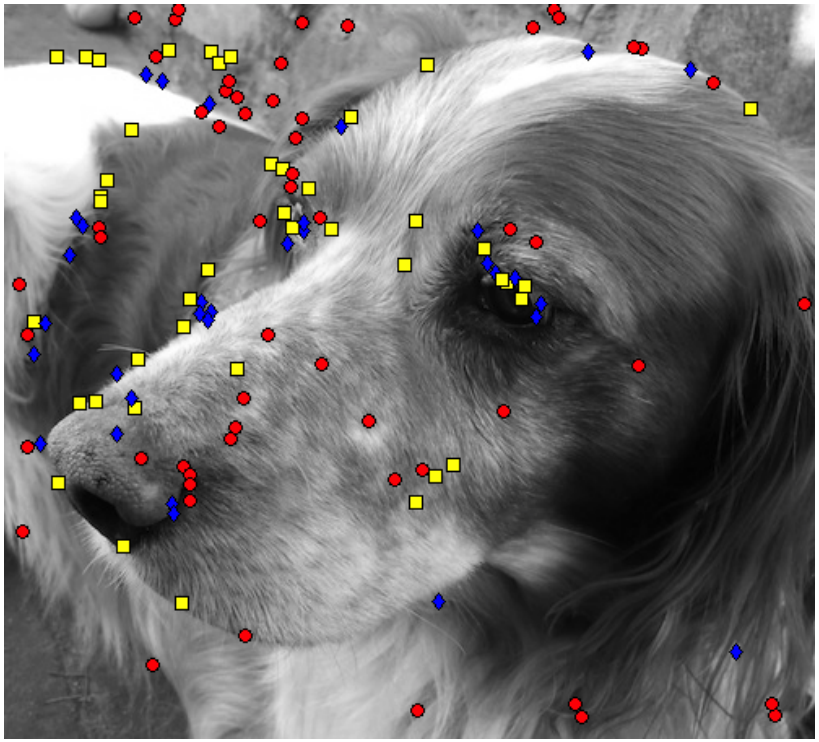
- Simplified workflow of the classification algorithm



Classification: adapting the algorithm

- We want the number of modes M, not thresholds for H
- Cannot rely directly on the number of Gaussians, since [9] over-segments and generates too many contributions
- Adjustments are required:
  - initialize $M = n + 1$ contributes of the mixture
  - determine the largest weight P_{max}
  - suppress $P_i,\ 1 \leq i \leq n+1$ such that $P_i \leq 0.2 \cdot P_{max}$
- Keypoint's neighborhood experimentally set to 32×32
  - large enough to respect the hypothesis of Gaussianity
  - small enough to include only local statistics

Classification: example



The proposed attack

SIFT countering: finding weak spots I

- A candidate keypoint must pass 3 checks
  - Distance from other candidate(s) $\geq Threshold_1$
  - Contrast $\geq Threshold_2$
  - Distance from edges $\geq Threshold_3$
- The idea behind SIFT countermeasures is to work on such checks

SIFT countering: finding weak spots II

- Common objectives of SIFT countering

> To alter the keypoint's neighborhood in such a way that at least one of the check fails

> To alter two matching descriptors in such a way that they do not match anymore

> Each manipulation comes at a rather high price in terms of quality. Visual degradation needs to be the lowest possible

SIFT countering: finding weak spots III

- Failing checks
  - DoG distance: introduce a fake candidate too close to the real candidate
  - contrast: locally modify the contrast until it drops below the threshold
  - "cornerness": can't move edges around!
- Failing the matching
  - modify the gradient orientations so that the descriptor is altered
  - attacking the matching algorithm (not related to SIFT)

The proposed attack: tools I – Gaussian Smoothing

- Blurs the image by convolving it with a Gaussian kernel
- Experimental results [3]: low variance (σ) can help reducing keypoints
  - could also add keypoints!
- Attenuates high frequencies, flattens values of a given region, reduces contrast which eventually drops below SIFT's threshold
- Parameters: window size $h = 3$, $\sigma = 0.70$

The proposed attack: tools II – Collage [1]

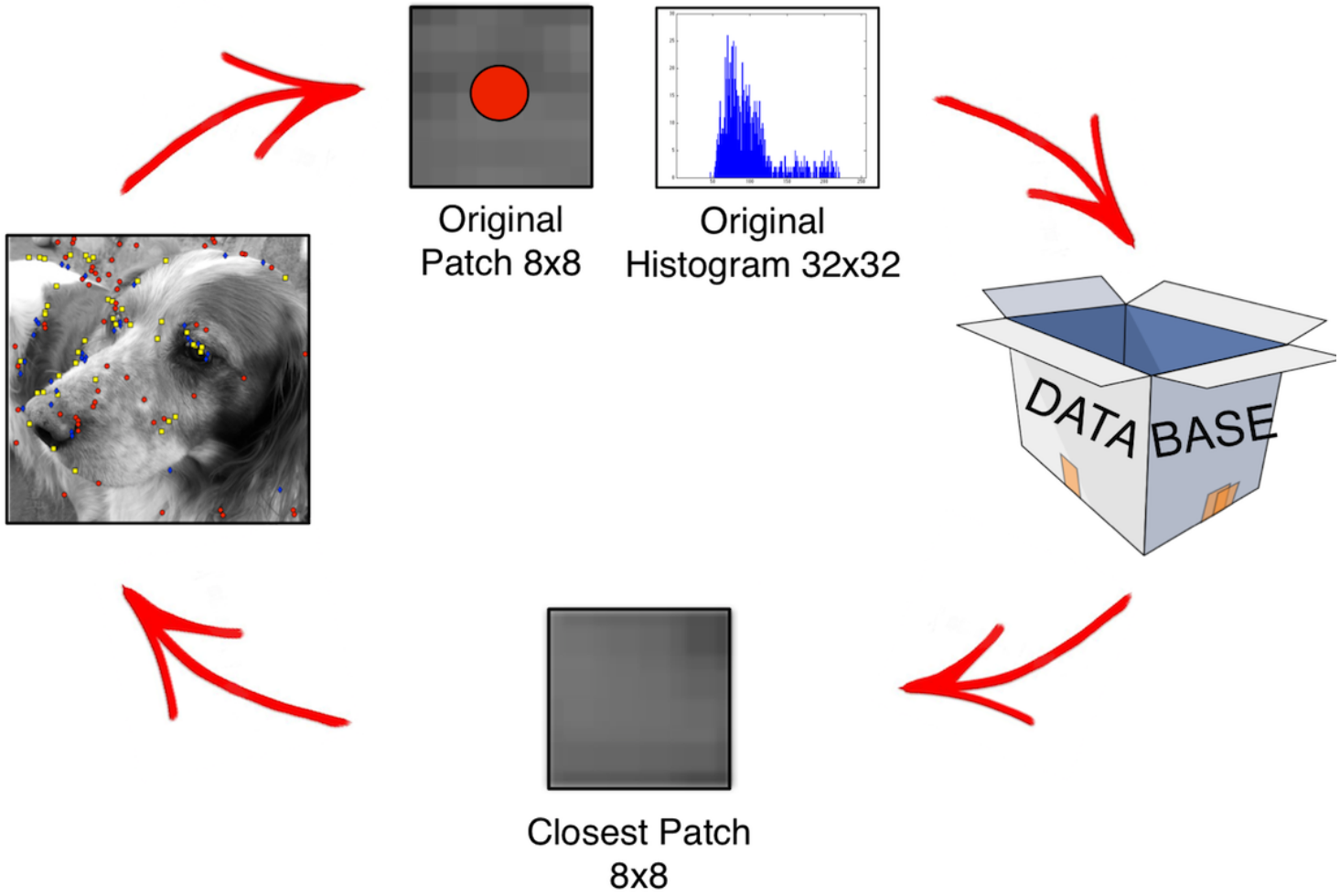
- Original patch replaced with the most similar *"keypoint–free"* patch of same size from a database
- Similarity metric: *histogram intersection distance* d_{int}

$$d_{int}(H_{orig}, H_{db}) = \frac{\sum_{j=1}^{L} min\big(H_{orig}(j), H_{db}(j) \big)}{\sum_{j=1}^{L} H_{db}(j)}$$

H_{orig}, H_{db} histograms of the original patch and of a patch stored in the database; j histogram bin; $L = 255$

$$d_{min} = \arg\min_{n} \Big[d_{int}(H_{orig}, H_{db}(n)) \Big]$$

The proposed attack: tools II – Collage [1]



The proposed attack: tools III – RMD [3]

- *Removal with Minimum Distortion*
- Calculates a small patch ϵ that added to the neighborhood of a keypoint allows its removal
- The coefficients of ϵ are chosen in such a way to:
  - reduce the contrast around the keypoint at the DoG level
  - invalidate the SIFT contrast check
  - locally introduce the minimum visual distortion
- In math: let $\mathbf{x} = (x, y, \sigma)$ be a keypoint and let $D(\mathbf{x})$ be the DoG in $\mathbf{x}$:

$$\epsilon = arg\ min_{\epsilon : D'(\mathbf{x}) = D(\mathbf{x}) + \delta} \left(||\epsilon||^2 \right)$$

The proposed attack: tools III – RMD

- δ controls the strength of the attack
  - $|D(\mathbf{x})|$ reduced by $|\delta|$ until it drops below contrast threshold
- The size of ϵ depends on the strength of the keypoint (in terms of spatial support)
  - the larger the support, the stronger (hence more visible) the attack
  - altered DoG is $D(x+u, y+v, \sigma)$
    - $(u, v) \in [-6\sqrt{2}h\sigma, 6\sqrt{2}h\sigma]$
    - $h = 2^{\frac{1}{3}}$

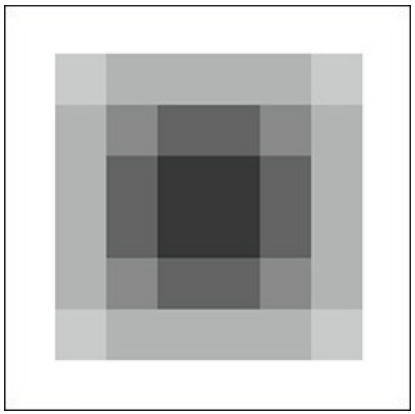
The proposed attack: tayloring the right attack

- *Smoothing* is effective on all the classes but . . .
  - ☺ reduces the keypoints without a significant loss of quality
  - ☹ not effective against *"harder"* keypoints, more powerful countermeasures required
- *Collage* is effective on uniform or noise-like regions which do not have many noticeable visual details (unimodal, multimodal)
- *RMD* is effective on patches containing geometric edges (bimodal)

The proposed attack: reducing the impact

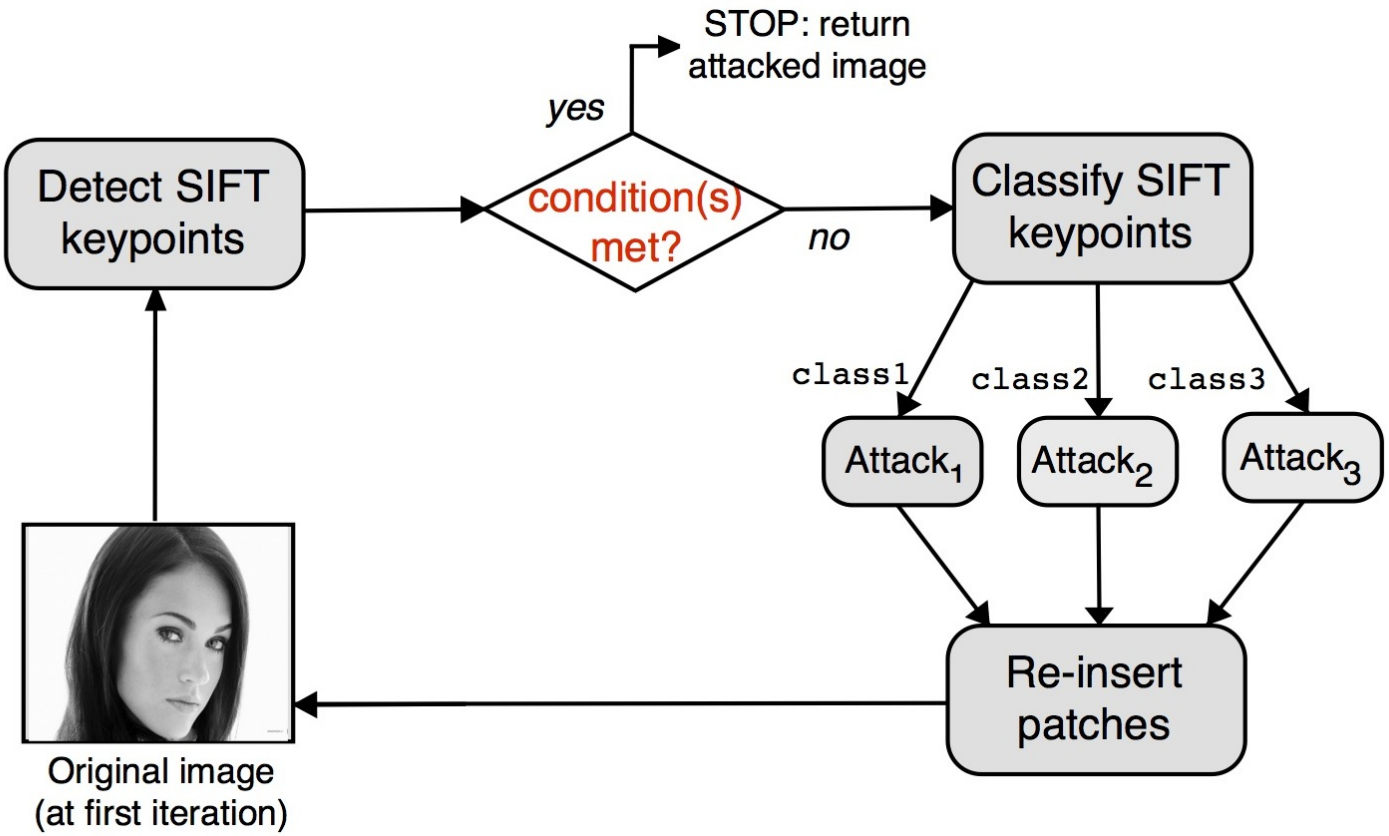
- Visible artifacts along the borders of the patches masked with a linear combination:

$$\mathbf{patch}_{atk} = W_8 \cdot \mathbf{patch}_{orig} + (1\text{-}W_8) \cdot \mathbf{patch}_{\{Smoothing/RMD/Collage\}}$$



$$W_8 = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 0.8 & 0.7 & 0.7 & 0.7 & 0.7 & 0.8 & 1.0 \\ 1.0 & 0.7 & 0.5 & 0.3 & 0.3 & 0.5 & 0.7 & 1.0 \\ 1.0 & 0.7 & 0.3 & 0.1 & 0.1 & 0.3 & 0.7 & 1.0 \\ 1.0 & 0.7 & 0.3 & 0.1 & 0.1 & 0.3 & 0.7 & 1.0 \\ 1.0 & 0.7 & 0.5 & 0.3 & 0.3 & 0.5 & 0.5 & 1.0 \\ 1.0 & 0.8 & 0.7 & 0.7 & 0.7 & 0.7 & 0.8 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix}$$

The proposed attack: framework

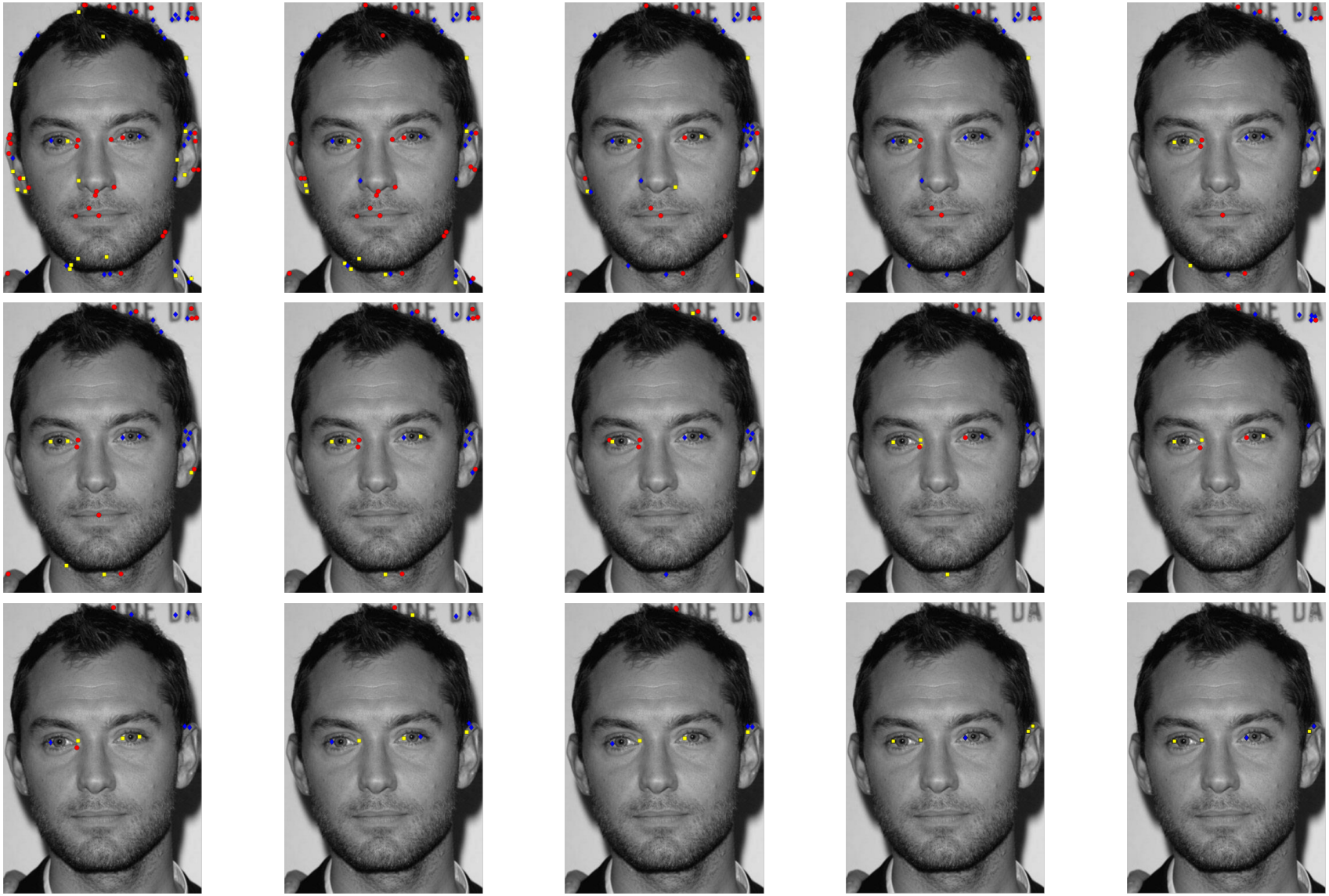


New keypoints introduced at iteration j are classified and attacked again at the following iteration $j+1$

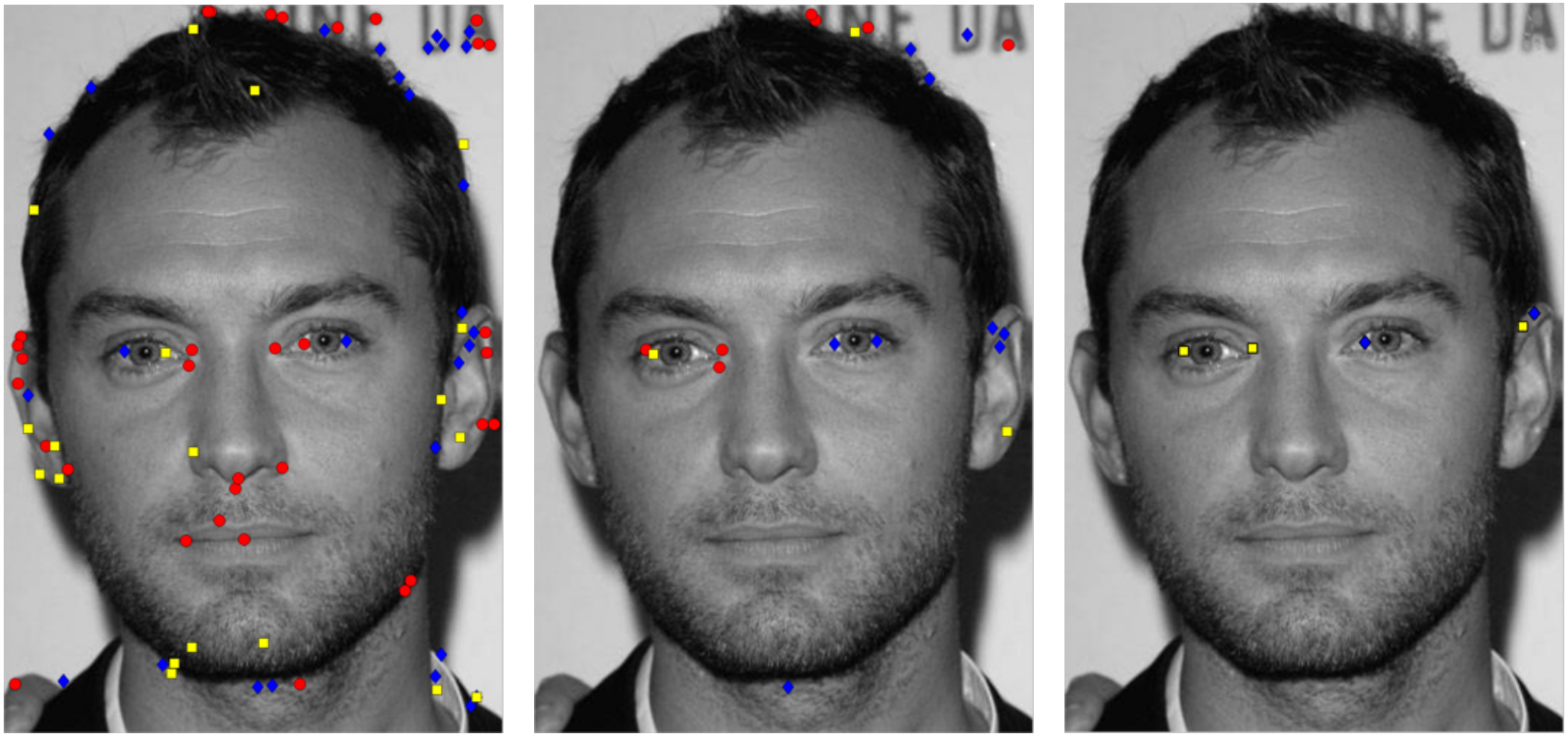
The proposed attack: pseudocode

```
1:  procedure ATTACK_SIFT(image, max_iter)
2:      j ← 1
3:      while (j ≤ max_iter or n_keypoints > 0) do
4:          keypoints = calculate_SIFT( image )
5:          kp_classes = classify_SIFT( keypoints )
6:          if j ≤ max_iter/2 then
7:              smoothing_attack(kp_classes)
8:          else
9:              collage_attack(unimodal,multimodal)
10:             RMD_attack(bimodal)
11:         end if
12:         j ← j + 1
13:     end while
14: end procedure
```

The proposed attack: example – odd iterations



The proposed attack: example – iterations 0,15,30



The proposed attack: example – iterations 0,30

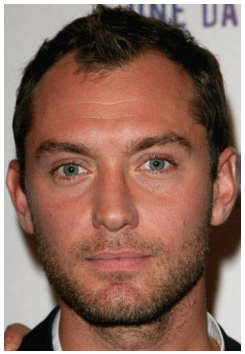
Experimental results

TO DO: three experiments

Experimental setup: image data set I

- We gathered 60 natural images
  - size ranging from 400×400 to 600×800 pixels
- We divided them into 3 subsets: *landscape, animal, faces*
- Posing an increasing difficulty to the system
  - **"Easy" – landscape**: many keypoints, easier removal due to high variance of the content (e.g. trees, houses)
  - **"Hard" – faces**: close shots, less keypoints but placed in critical locations (e.g. eyes, nose, mouth)
  - **"Medium" – animals**: midway between the two "extrema"

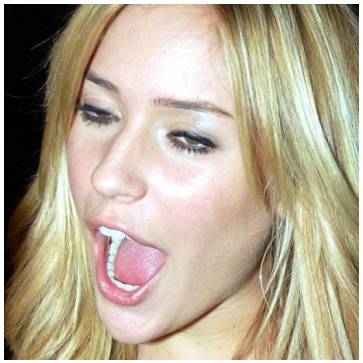
Experimental setup: image data set II



face1.jpg

face2.jpg

face3.jpg



face4.jpg

face5.jpg

animal1.jpg

animal2.jpg

animal3.jpg

animal4.jpg

animal5.jpg

landscape1.jpg

landscape2.jpg

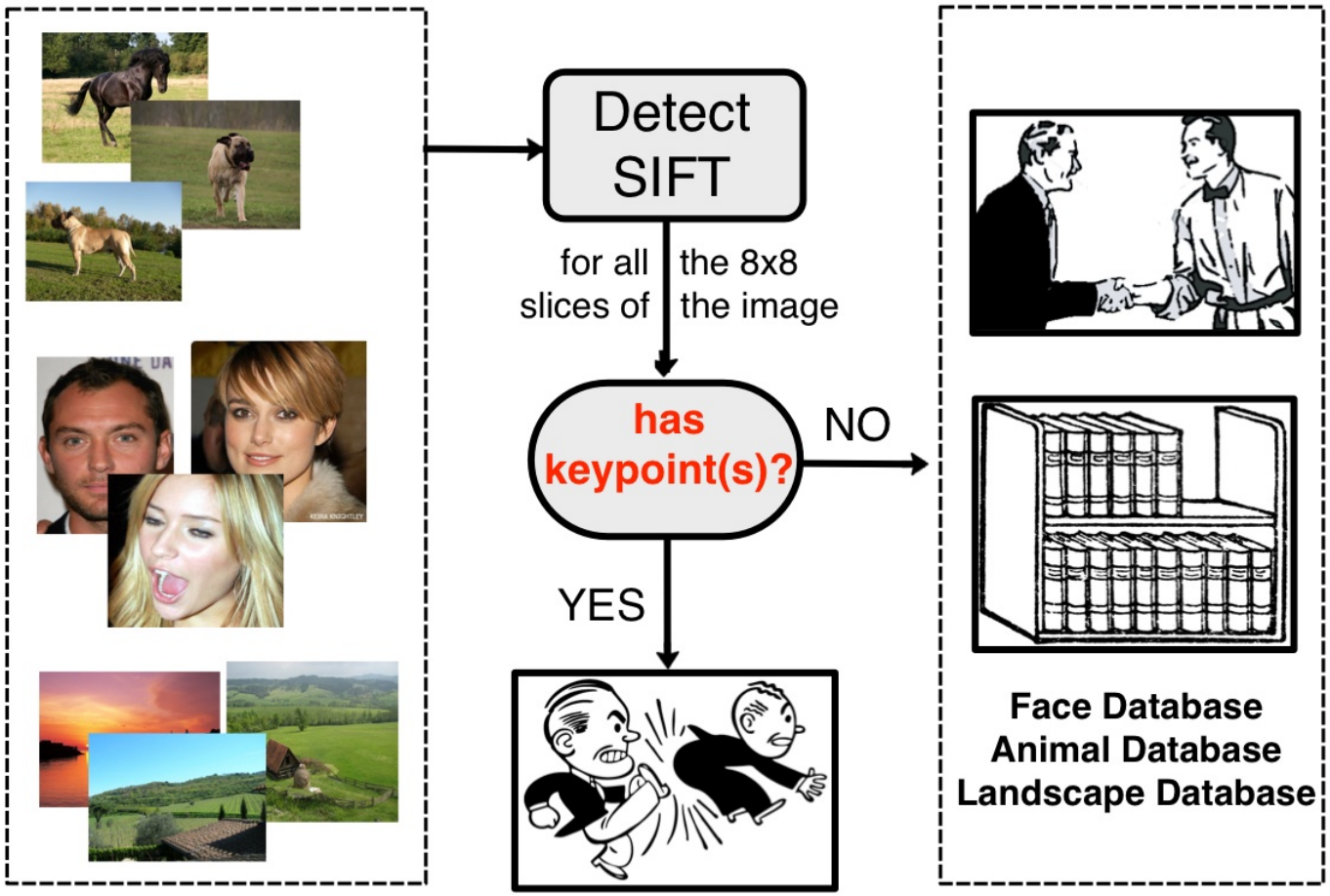
landscape3.jpg

landscape4.jpg

landscape5.jpg

Experimental setup: collage database

- 60 images (20 for each content group) to build the "keypoint-free" collage database



Advantages of classification: procedure

Does the classification bring any advantages at all?

- Fixing some parameters:
  - 15 images
  - decreasing removal rate $\in [95\%, 90\%, 85\%, 80\%]$
  - maximum number of iterations ($max\_iter = 50$)
- Iterating separately: *Classification-based*, *RMD*, *Collage*, *Smoothing* attacks until last iteration or desired removal
  - reaching removal = success = ☺
- Averaging results only on successful images
  - effective removal rate at halting
  - average patch PSNR (local quality)
  - average number of iterations

Advantages of classification: results I

	Attack	Successful images (success rate)	Average effective removal rate	Average patch PSNR (dB)	Average number of iterations
95% removal	**Proposed**	**14/15 (93%)**	**98.3%**	**42.32**	**27**
	RMD	15/15 (100%)	96.0%	26.71	16
	Collage	3/15 (20%)	96.8%	39.94	18
	Smoothing	0/15 (0%)	×	×	×
90% removal	**Proposed**	**15/15 (100%)**	**92.7%**	**43.79**	**22**
	RMD	15/15 (100%)	91.9%	28.42	10
	Collage	10/15 (67%)	90.8%	36.42	10
	Smoothing	0/15 (0%)	×	×	×
85% removal	**Proposed**	**15/15 (100%)**	**85.7%**	**44.51**	**19**
	RMD	15/15 (100%)	87.8%	29.41	9
	Collage	12/15 (80%)	86.2%	35.82	8
	Smoothing	0/15 (0%)	×	×	×
80% removal	**Proposed**	**15/15 (100%)**	**82.3%**	**45.13**	**17**
	RMD	15/15 (100%)	82.0%	30.50	7
	Collage	15/15 (100%)	82.2%	33.41	9
	Smoothing	0/15 (0%)	×	×	×

Advantages of classification: results II

- No attack reaches perfect removal (100%) within 50 iterations
- *Classification-based Attack* outperforms *RMD*
  - comparable success rate and effective removal
  - significant gain in local visual quality (avg. patch PSNR +15.6dB)
- *Smoothing* attack unable to meet even the lowest requirement (80%)
  - first few iterations reduce keypoints, following ones increase them back to (or even above) their original value
  - best result: 72.4% with average patch PSNR of 39dB

Advantages of classification: visual examples

- Visual comparison of the three most effective methods

Original

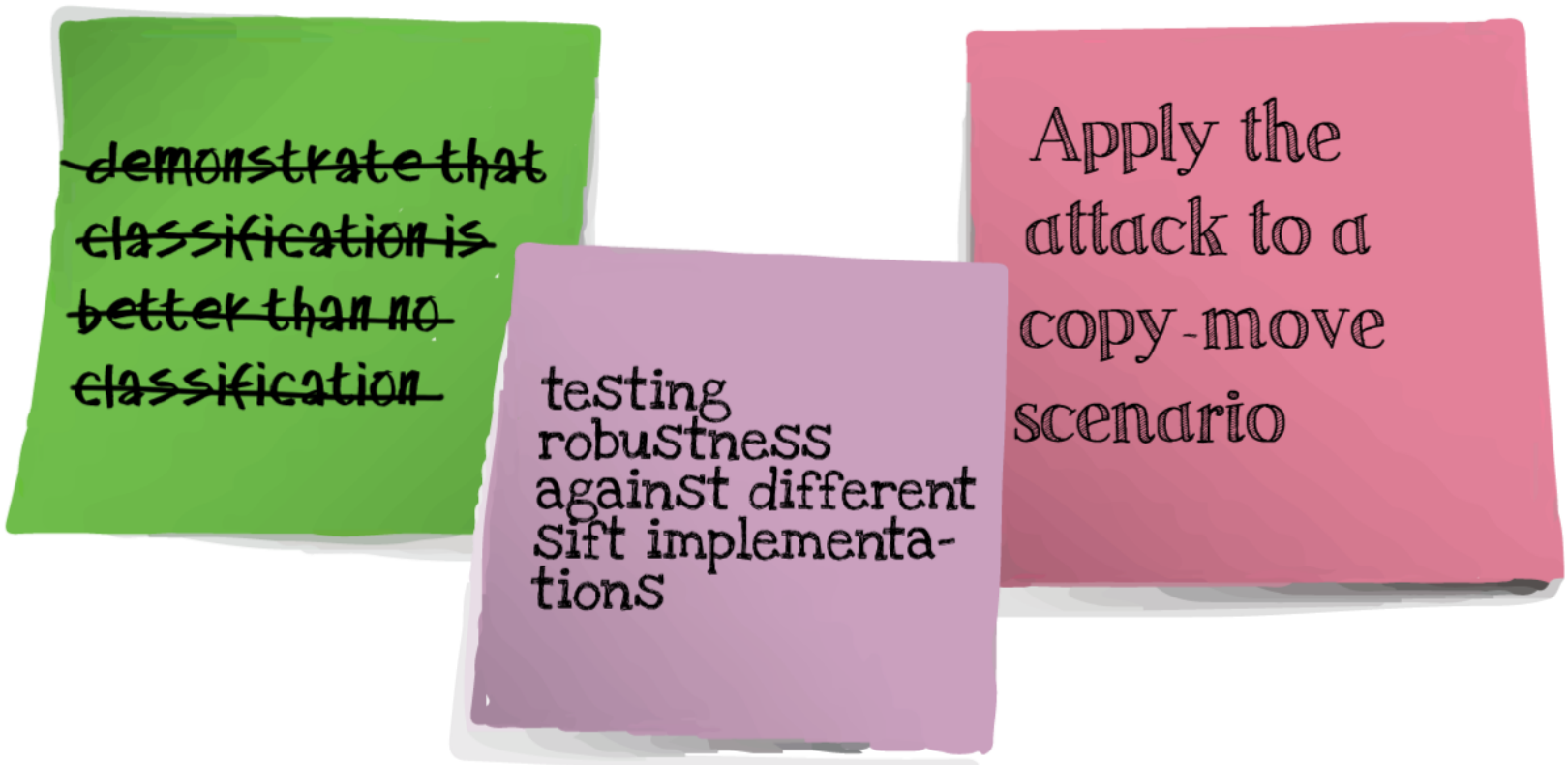
Proposed:
98%, 37.8dB

RMD:
94%, 25.6dB

Collage:
91%, 35.1dB

- Artifacts of *RMD* (large uniform "dots") and *Collage* more noticeable than those of *Classification-based Attack*

TO DO: two experiments



Test of robustness: SIFT implementations I

- There exist a considerable number of SIFT implementations (majority is open source)
  - very popular algorithm on many fields
  - the original SIFT is patented (demo only)
  - people use a lot of programming/scripting languages
- Different implementations = different results
  - number and coordinates of keypoints
  - heterogeneous tweakable parameters
  - approaches to tasks (computing DoGs, smoothing . . .)
  - speed of execution

Test of robustness: SIFT implementations II

What does happen when an image cleaned with a SIFT implementation, is analyzed with a totally different one?

- We tested the attack with 4 versions of SIFT
  1. `VLFeat` [10]: "reference library" for the SIFT-based (counter-) forensics. Written in C language, Matlab/Python interfaces
  2. `Matlab/C SIFT` [11]: predecessor of `VLFeat`, thus superseded but still used (see [5])
  3. `RobHess` [12] library. Written in C on well-established OpenCV computer vision library
  4. `JIFT` [13] library. Written in C on VXL computer vision library

Test of robustness: experimental settings

- Keypoints are removed by means of the proposed *Classification-based Attack* supported by `VLFeat`
- Results are analyzed with the 4 detectors
- "Used as they are", no tuning no cheating!

	Octaves			Thresholds		
	Number	Initial	Intervals	Contrast	Edge	Peak
`VLFeat`	1	0	3	×	10	4
`Matlab/C`	1	0	3	0.03	10	4
`RobHess`	1	0	3	0.04	10	0.8
`JIFT`	1	0	3	0.03	10	0.8

- *max_iter* = 30 (from previous experiment), 15 images, work only on first octave

Test of robustness: results I

- VLFeat-based attack VS MATLAB/C-based and JIFT-based detection: ☺

SIFT:	VLFEAT [10]			MATLAB/C [11]			JIFT (OpenCV) [13]		
Image	Before	After	Removal	Before	After	Removal	Before	After	Removal
face_1	90	3	**96.7%**	167	36	**78.4%**	50	7	**86.0%**
face_2	58	2	**96.6%**	188	41	**78.2%**	69	3	**95.7%**
face_3	53	8	**81.1%**	131	26	**80.2%**	131	9	**93.1%**
face_4	159	10	**93.7%**	267	41	**84.6%**	99	11	**88.9%**
face_5	118	2	**98.3%**	618	152	**75.4%**	149	13	**91.3%**
land_1	36	5	**86.1%**	213	32	**85.0%**	42	11	**73.8%**
land_2	183	11	**94.0%**	397	82	**79.4%**	154	5	**96.8%**
land_3	386	41	**89.4%**	478	143	**70.1%**	968	96	**90.1%**
land_4	432	36	**91.7%**	454	121	**73.3%**	1069	156	**85.4%**
land_5	192	12	**93.8%**	711	186	**73.8%**	166	6	**96.4%**
anim_1	417	19	**95.4%**	439	156	**71.1%**	207	48	**76.8%**
anim_2	95	2	**97.9%**	332	95	**71.4%**	26	8	**69.2%**
anim_3	104	3	**97.1%**	709	246	**65.3%**	35	6	**82.9%**
anim_4	443	30	**93.2%**	536	148	**72.4%**	319	20	**93.7%**
anim_5	112	9	**92.0%**	435	190	**56.3%**	69	7	**89.9%**
Average:			**93%**			**75%**			**87.3%**

Test of robustness: results II

- VLFeat-based attack VS RobHess-based detection:☹

SIFT:	VLFEAT [10]			ROBHESS [12]		
Image	Before	After	Removal	Before	After	Removal
face_1	90	3	**96.7%**	47	45	**4.3%**
face_2	58	2	**96.6%**	20	16	**20.0%**
face_3	53	8	**81.1%**	26	30	**0%**
face_4	159	10	**93.7%**	61	57	**6.6%**
face_5	118	2	**98.3%**	80	66	**17.5%**
land_1	36	5	**86.1%**	30	30	**0%**
land_2	183	11	**94.0%**	65	66	**0%**
land_3	386	41	**89.4%**	118	100	**15.3%**
land_4	432	36	**91.7%**	148	149	**0%**
land_5	192	12	**93.8%**	78	83	**0%**
anim_1	417	19	**95.4%**	137	147	**0%**
anim_2	95	2	**97.9%**	46	44	**4.3%**
anim_3	104	3	**97.1%**	57	55	**3.5%**
anim_4	443	30	**93.2%**	125	125	**0%**
anim_5	112	9	**92.0%**	58	60	**0%**
Average:			**93%**			**2.5%**

- Why? Because RobHess finds a lot of different keypoints on which the attack was clearly not carried out

Test of robustness: new results I

- Fixing the problem: combine `VLFeat` and `RobHess` during the classification-based attack

SIFT:	VLFEAT [10]			MATLAB/C [11]			JIFT (OpenCV) [13]		
Image	Before	After	Removal	Before	After	Removal	Before	After	Removal
face_1	90	3	**96.7%**	167	39	**76.6%**	50	9	**82.0%**
face_2	58	1	**98.3%**	188	41	**78.2%**	69	3	**95.7%**
face_3	53	1	**98.1%**	131	26	**80.2%**	131	9	**93.1%**
face_4	159	12	**92.5%**	267	37	**86.2%**	99	8	**92.0%**
face_5	118	6	**95.0%**	618	161	**74.0%**	149	13	**92.3%**
land_1	36	6	**83.3%**	213	40	**81.2%**	42	11	**73.8%**
land_2	183	4	**97.8%**	397	81	**79.6%**	154	3	**98.0%**
land_3	386	52	**86.5%**	478	151	**68.4%**	968	96	**90.1%**
land_4	432	44	**89.8%**	454	107	**76.4%**	1069	156	**95.9%**
land_5	192	10	**94.8%**	711	194	**72.7%**	166	6	**96.4%**
anim_1	417	18	**95.7%**	439	162	**69.9%**	207	48	**76.8%**
anim_2	95	4	**95.8%**	332	104	**68.7%**	26	8	**69.2%**
anim_3	104	7	**93.3%**	709	255	**64.0%**	35	6	**82.9%**
anim_4	443	35	**92.1%**	536	145	**73.0%**	319	20	**93.8%**
anim_5	112	8	**92.9%**	435	187	**57.0%**	69	7	**89.9%**
Average:			**93.5%**			**74%**			**87.4%**

- No significant changes for the detectors above ☺

Test of robustness: new results II

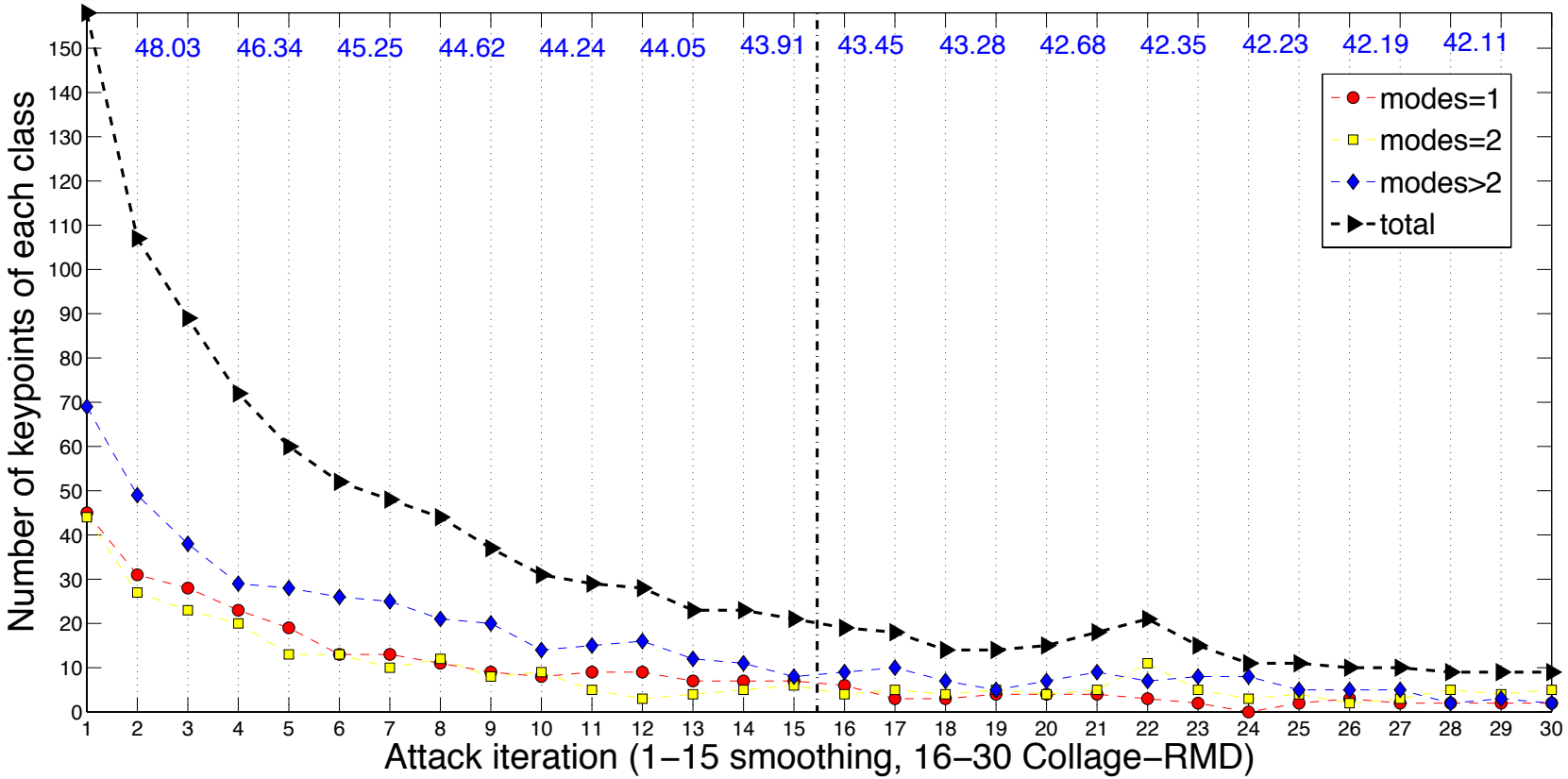
- VLFeat+RobHess-based attack VS RobHess-based detection: ☺

SIFT:	VLFEAT [10]			ROBHESS [12]		
Image	Before	After	Removal	Before	After	Removal
face_1	90	3	**96.7%**	47	7	**85.1%**
face_2	58	1	**98.3%**	20	5	**75.0%**
face_3	53	1	**98.1%**	26	4	**84.6%**
face_4	159	12	**92.5%**	61	5	**91.8%**
face_5	118	6	**95.0%**	80	10	**87.5%**
land_1	36	6	**83.3%**	30	12	**60.0%**
land_2	183	4	**97.8%**	65	11	**83.1%**
land_3	386	52	**86.5%**	118	29	**75.4%**
land_4	432	44	**89.8%**	148	43	**70.1%**
land_5	192	10	**94.8%**	78	16	**79.5%**
anim_1	417	18	**95.7%**	137	34	**75.2%**
anim_2	95	4	**95.8%**	46	8	**82.6%**
anim_3	104	7	**93.3%**	57	11	**80.7%**
anim_4	443	35	**92.1%**	125	33	**73.6%**
anim_5	112	8	**92.9%**	58	11	**81.0%**
Average:			**93.5%**			**79%**

- Dramatically more effective: 79% vs 2.5%!

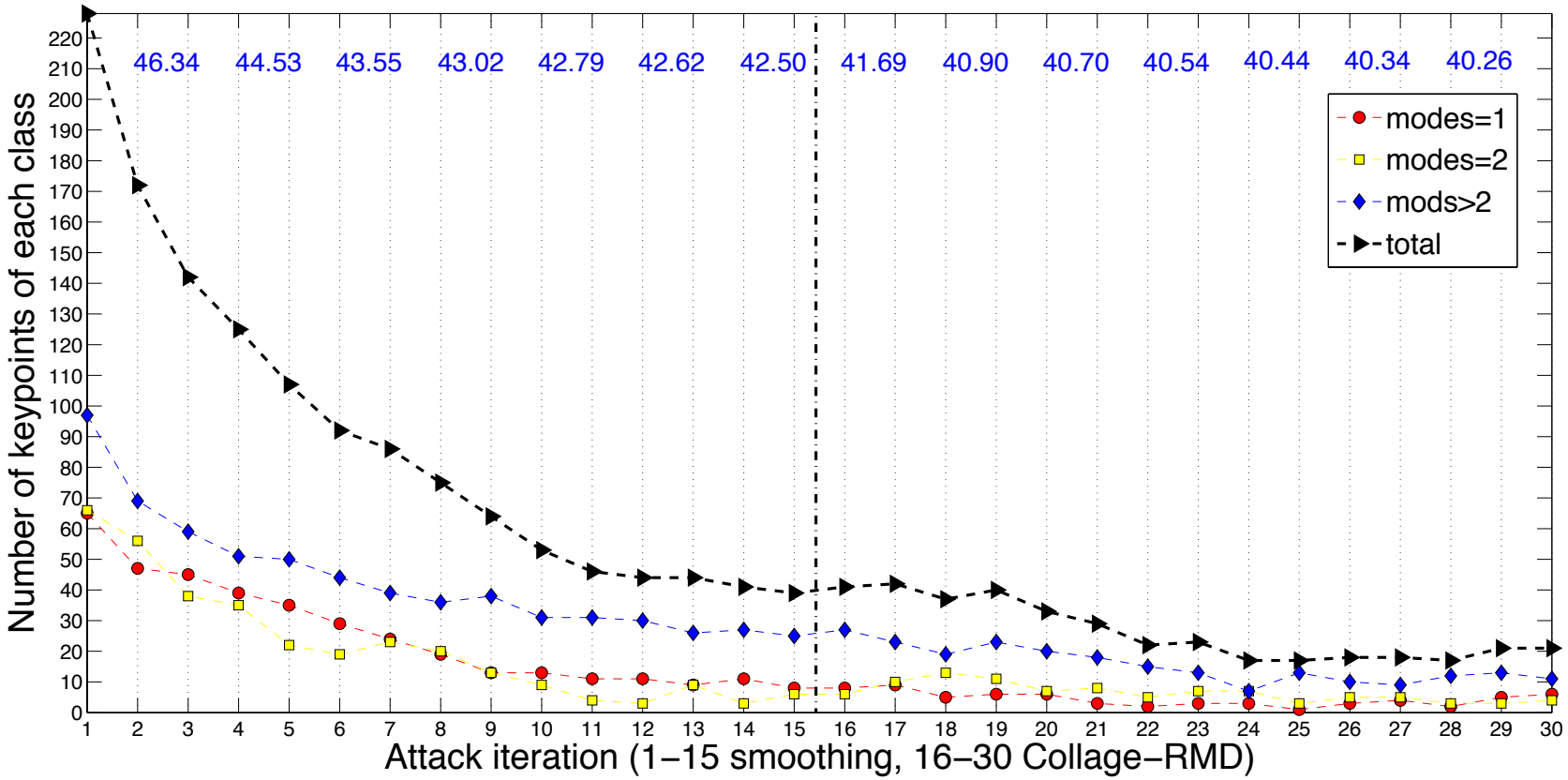
Test of robustness: VLFeat only plots

- Number and classes of keypoints for each iteration of the attack on test image *face4.jpg*: `VLFeat` alone



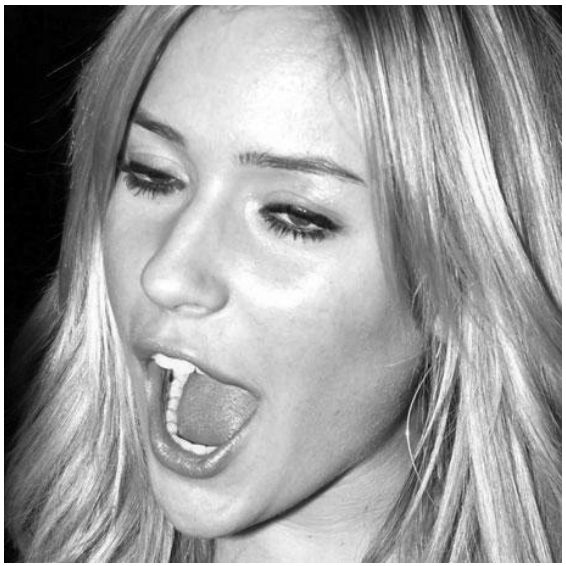
Test of robustness: VLFeat+RobHess plots

- Number and classes of keypoints for each iteration of the attack on test image *face4.jpg*: `VLFeat+RobHess`

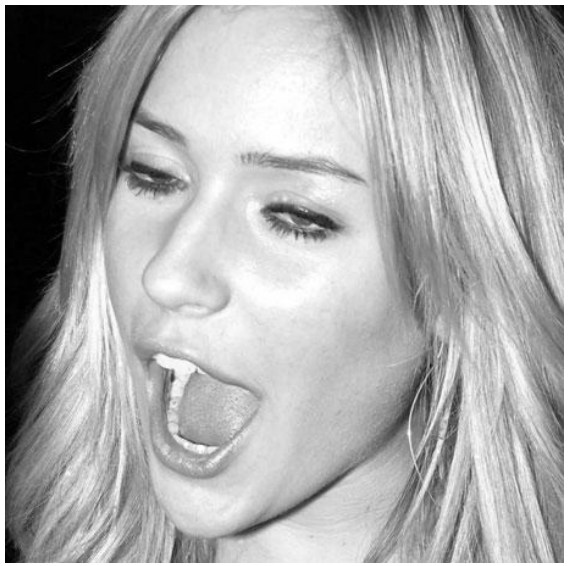


Test of robustness: visual example

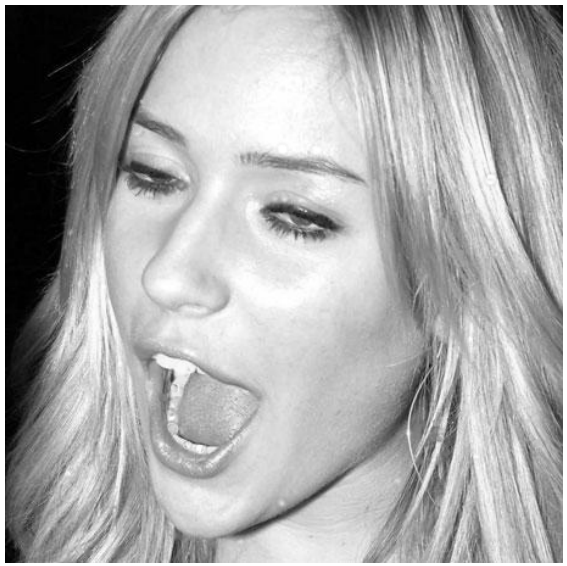
- Quality loss of `VLFeat+RobHess` barely noticeable with respect to `VLFeat` only
  - **40.26** dB vs **42.11** dB PSNR full frame
  - **34.02** dB vs **36.51** dB PSNR patch average
- Example of *face4.jpg*



Original

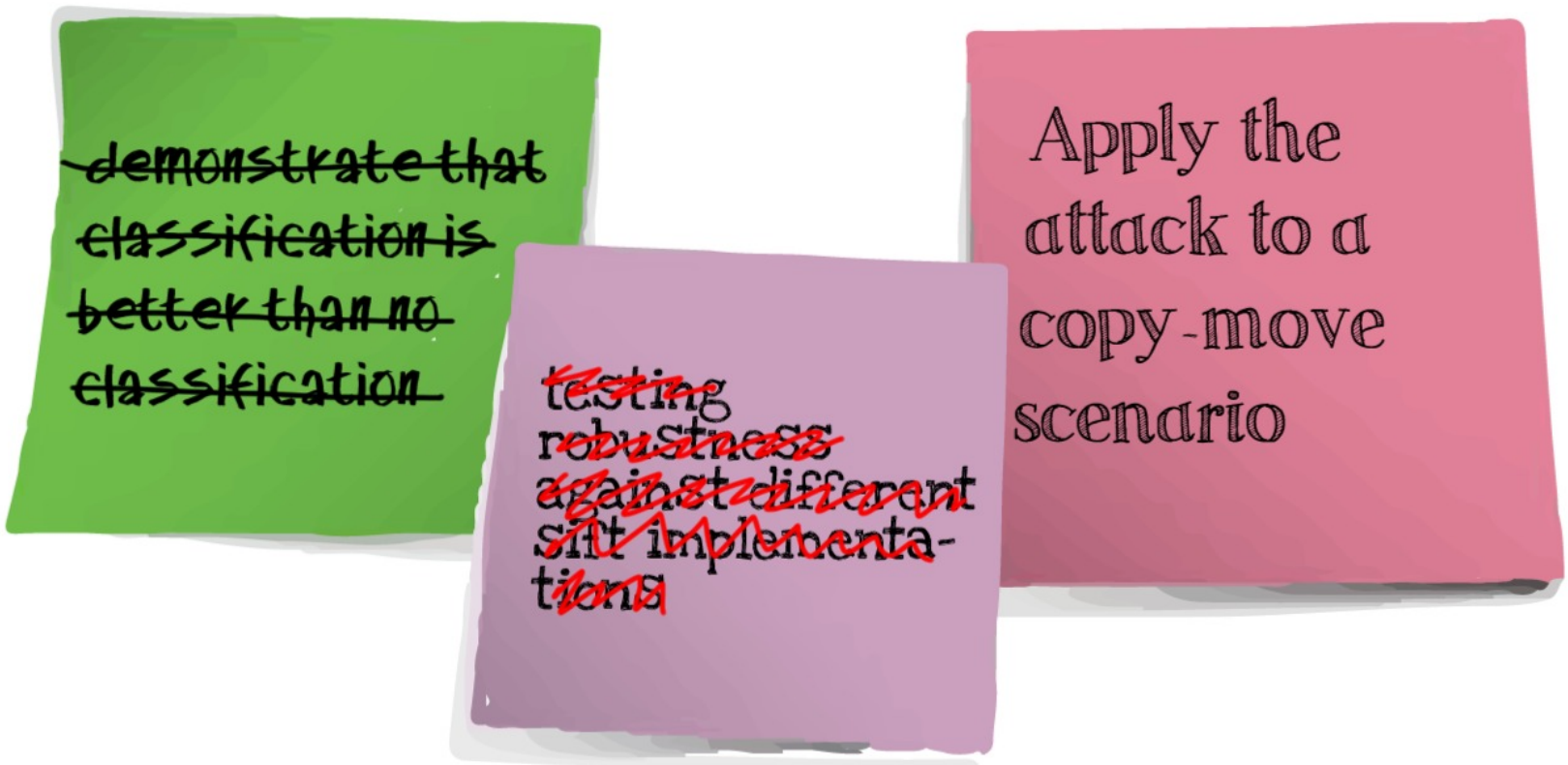


VLFeat only



VLFeat+RobHess

TO DO: one experiment



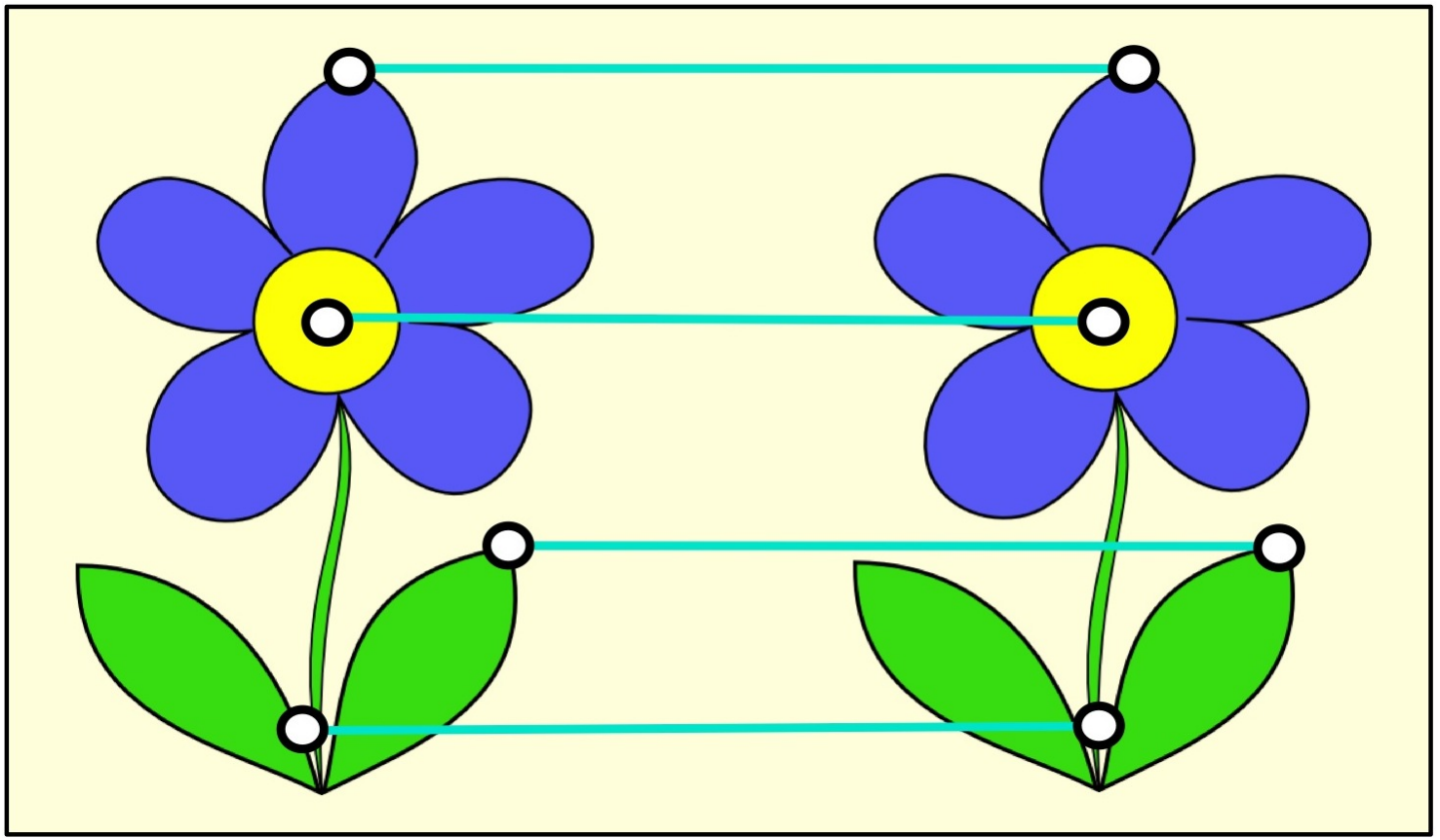
Copy-move scenario: introduction I

Can we hide the traces of copy-move forgery and invalidate a SIFT-based detector?

- Assumption: to work only on two copy-moved regions
  - for the sake of clarity, no loss of generality
- New objective: to work only on source and destination patches, rather than on whole image

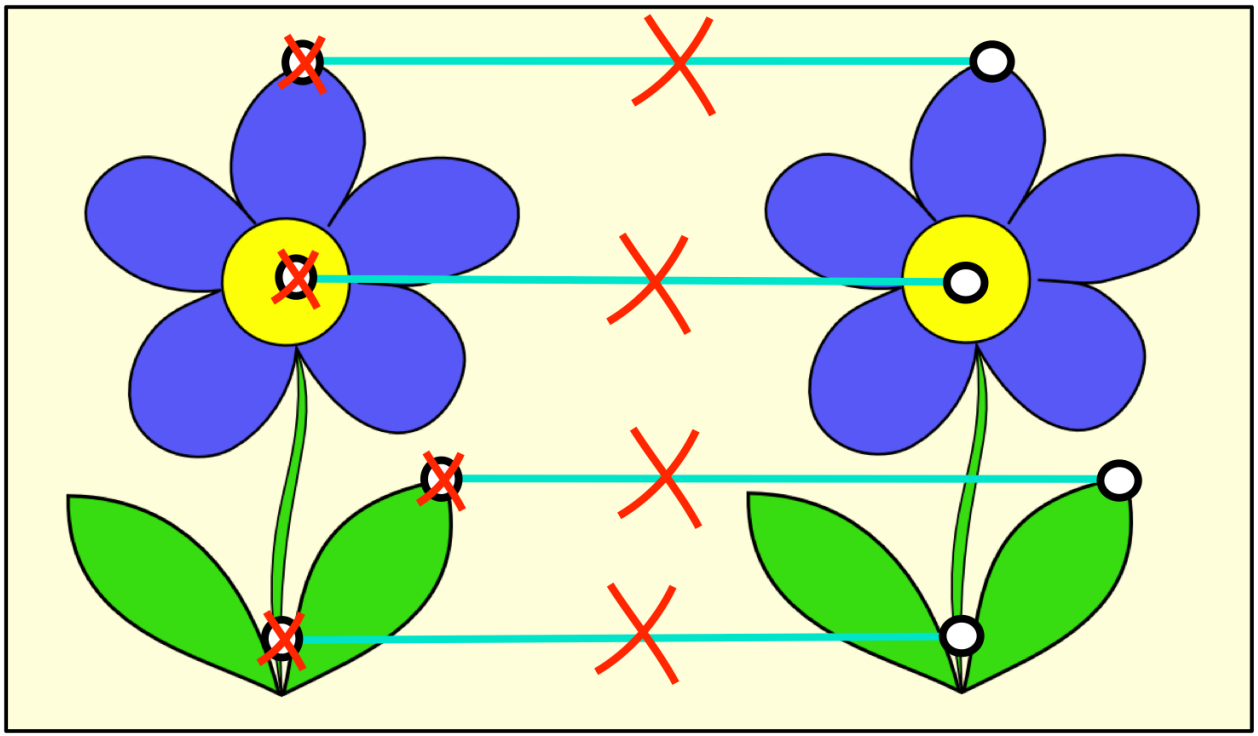
Copy-move scenario: introduction II

- Copy-move forgery is detected by matching pairs of SIFT keypoints between similar regions
  - if there are enough matches (usually≥ 3) then tampering



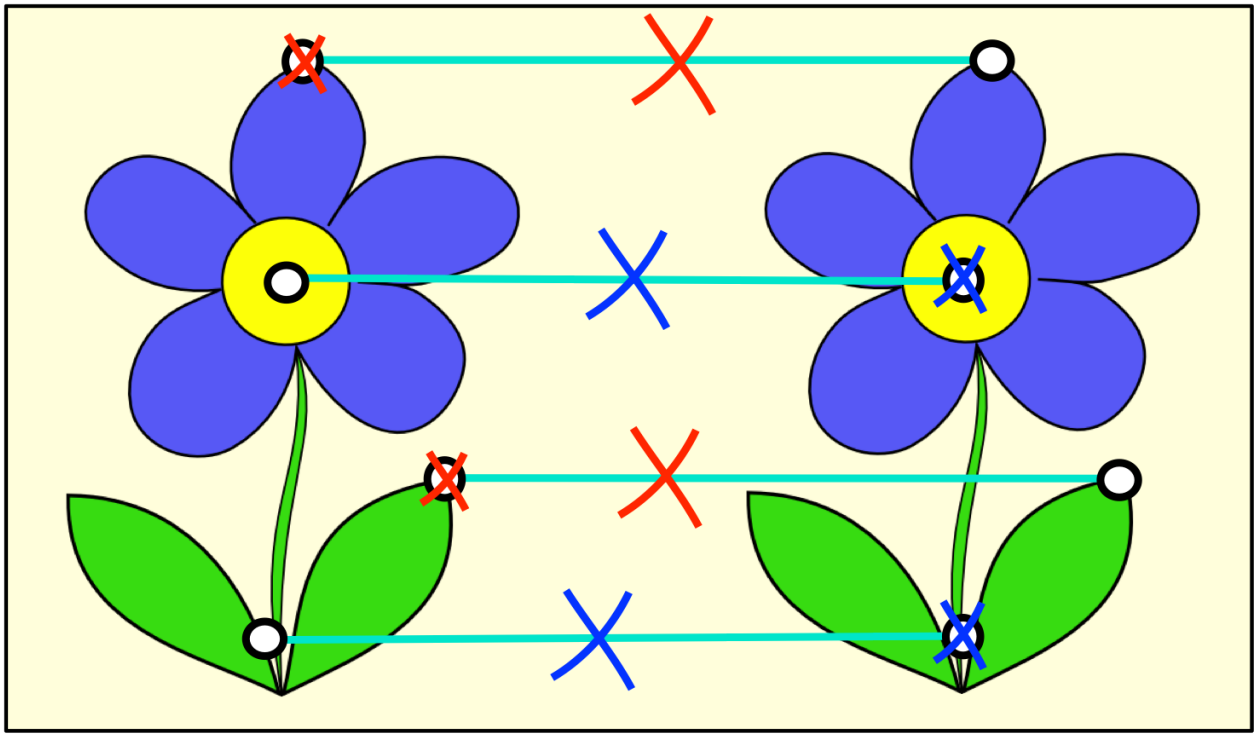
Copy-move scenario: introduction II

- To destroy a match it is not necessary to delete both keypoints, one is enough
  - ☹ one region is clean and the other has all keypoints



Copy-move scenario: introduction II

- Evenly spread the attack to reduce the visual impact of the hiding procedure
  - ☺ remove 1/2 of the matching keypoints from each region



Copy-move scenario: hiding procedure

- Copy-move detected with the method described in [8]
- The matching regions, R_1 and R_2 are attacked with the *Classification-based Attack*
  - $1/2$ of the matches are destroyed by deleting the member of the match belonging to R_1
  - $1/2$ by deleting the member belonging to R_2
- Minor variants
  - $max\_iter = 40$ (before: 30)
  - last 10 iterations of the attack with adaptive parameters
    - stronger the keypoint, stronger the parameters of *Collage/RMD*

Copy-move scenario: image data set

- 10 images containing a copy-move manipulation

Copy-move scenario: results I

- Repeating the attack up to 1000 times to understand whether perfect removal can be achieved

		Smoothing		Collage		RMD[3]		Proposed	
image	Before	Iteration	After	Iteration	After	Iteration	After	Iteration	After
image_1	53	1000	14	1000	6	1000	3	**183**	**0**
image_2	49	1000	15	1000	8	1000	9	**91**	**0**
image_3	12	1000	3	**10**	**0**	13	0	23	0
image_4	55	1000	9	1000	2	1000	10	**161**	**0**
image_5	50	1000	9	1000	3	1000	2	**41**	**0**
image_6	34	1000	2	39	0	39	0	**29**	**0**
image_7	43	1000	11	1000	5	**29**	**0**	62	0
image_8	19	1000	5	1000	1	53	0	**24**	**0**
image_9	109	1000	24	1000	12	1000	2	**520**	**0**
image_10	195	1000	56	1000	19	1000	26	**338**	**0**

- Outperforming the 3 single attacks, deleting all the matches in a low number of iterations

Copy-move scenario: results II

- The curves of *Classification-based Attack* (star markers) are below those of the single attacks

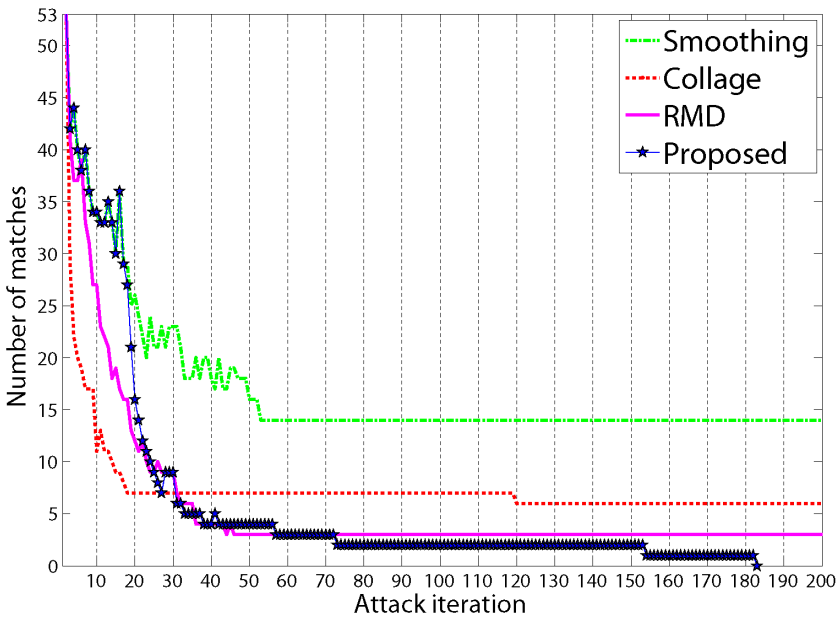


image_1

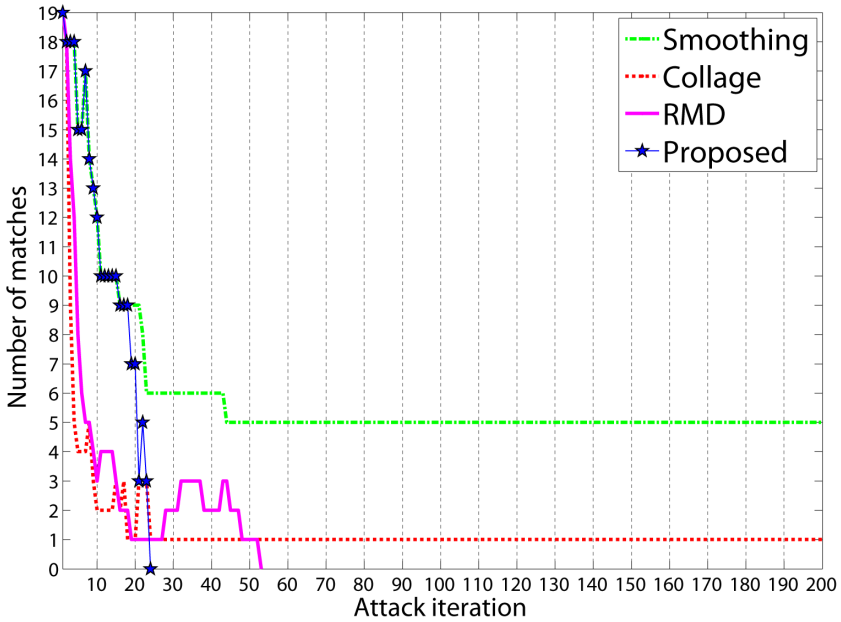


image_8

Copy-move scenario: visual example

- Detail of *image_5* (method VS iterations VS matches)

Smoothing
(iter = 1000, *M* = 9*)*

Collage
(iter = 1000, *M* = 3*)*

RMD
(iter = 1000, *M* = 2*)*

Proposed
(iter = 41, *M* = 0*)*

Computational burden

- Main contribution from cycling through all the keypoints during each of the iterations of the basic attack
- SIFT detection's time generally negligible
- On a World War II pc: 2GHz dual, 4GB RAM, 32bit OS
  - `VLFeat`: 0.5s; `Matlab/C`: 0.97s; `RobHess`: 0.68s; `JIFT`: 9s
- Full attack on a 600×450 image (386 keypoints)
  - 94s: 43s for iterations 1-15 (simple *Smoothing*) and 51s for iterations 16-30

The long road ahead

- Integrating all the detectors into the attack procedure
- Turning the attack into an usable tool / demo
- Injecting fake keypoints into the cleaned image
  - no keypoints are suspicious
- Applying injection to copy-move
  - replacing real matchings with false ones
- Applying the attack to a CBIR search engines
  - removal to disable them
  - injection to force false positives

The long road ahead: towards CBIR scenarios I

This image has copyright

The analyst wants to find out if a database (the web) contains unauthorized copies

This is a SIFT-based search engine. Its database (the web) contains some of our illegal copies

Before we redistributed our illegal copies all over the web after we removed all their SIFT keypoints

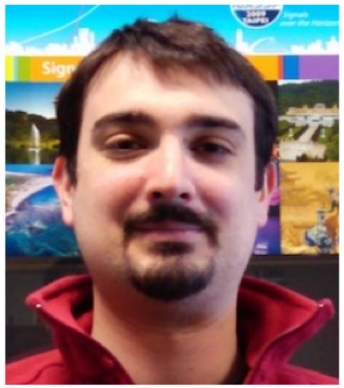
We have successfully disabled a copyright infringement

The long road ahead: towards CBIR scenarios II

Someone wants to find other versions similar to this input image

This is a SIFT-based search engine

We populated the database with random images first cleaned with our attack and then filled with false keypoints matching the input image



The wrong image is returned

We have successfully disabled an image retrieval engine

References I

Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei.

Secure and robust SIFT.

In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 637–640, 2009.

Thanh-Toan Do, Ewa Kijak, Teddy Furon, and Laurent Amsaleg.

Understanding the security and robustness of SIFT.

In *Proceedings of the international conference on Multimedia*, MM '10, pages 1195–1198, 2010.

Thanh-Toan Do, Ewa Kijak, Teddy Furon, and Laurent Amsaleg.

Deluding image recognition in SIFT-based cbir systems.

In *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, MiFor '10, pages 7–12, 2010.

Thanh-Toan Do, Ewa Kijak, Laurent Amsaleg, and Teddy Furon.

Enlarging hacker's toolbox: deluding image recognition by attacking keypoint orientations.

In *37th International Conference on Acoustics, Speech, and Signal Processing, ICASSP'12*, Kyoto, Japan, March 2012.

Roberto Caldelli, Irene Amerini, Lamberto Ballan, Giuseppe Serra, Mauro Barni, and A. Costanzo.

On the effectiveness of local warping against SIFT-based copy-move detection.

In *Proc. of Int'l Symposium on Communications, Control and Signal Processing (ISCCSP)*, Roma, Italy, May 2012.

D. G. Lowe.

Distinctive image features from scale-invariant keypoints.

Int.'l Journal of Computer Vision, 60(2):91–110, 2004.

References II

Sevinc Bayram, Husrev T. Sencar, and Nasir Memon.

A survey of copy-move forgery detection techniques.
In *Proc. of IEEE Western New York Image Processing Workshop*, 2008.

I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra.

A SIFT-based forensic method for copy move attack detection and transformation recovery.
Information Forensics and Security, IEEE Transactions on, 6(3):1099 –1110, sept. 2011.

Multi-modal gray-level histogram modeling and decomposition.

Image and Vision Computing, 20(3):203 – 216, 2002.

A. Vedaldi and B. Fulkerson.

VLFeat: An open and portable library of computer vision algorithms.
`http://www.vlfeat.org/`, 2008.

A. Vedaldi.

An open implementation of the SIFT detector and descriptor.
Technical Report 070012, UCLA CSD, 2007.

R. Hess.

An open-source SIFTlibrary.
In *Proceedings of the international conference on Multimedia*, pages 1493–1496. ACM, 2010.

J. Liu.

C++ implementation of scale-invariant feature transformation (SIFT).
Technical report, University of Manchester, 2005.