SIFT keypoint removal and injection for countering matching-based Image Forensics

Irene Amerini
Media Integration and Communication Center
University of Florence
Firenze, Italy
irene.amerini@unifi.it

Mauro Barni
Department of Information Engineering
University of Siena
Siena, Italy
barni@dii.unisi.it

Roberto Caldelli *
Media Integration and Communication Center
University of Florence
Firenze, Italy
roberto.caldelli@unifi.it

Andrea Costanzo
Department of Information Engineering
University of Siena
Siena, Italy
andreacos82@gmail.com

ABSTRACT

Scale Invariant Feature Transform (SIFT) has been widely employed in several image application domains, including Image Forensics (e.g. detection of copy-move forgery or near duplicates). Until now, the research community has focused on studying the robustness of SIFT against legitimate image processing, but rarely concerned itself with the problem of SIFT security against malicious procedures. Recently, a number of methods allowing to remove SIFT keypoints from an original image have been devised. Although quite effective, such methods produce an attacked image with very few (or no) keypoints, thus leaving cues that can be easily exploited by a forensic analyst to reveal the occurred manipulation. In this paper, we explore the topic of reintroducing fake SIFT keypoints into a previously cleaned image in order to address the main weakness of the existing removal attacks. In particular, we evaluate the fitness of locally adaptive contrast enhancement methods to the task of injecting new keypoints. The results we obtained are encouraging: (i) it is possible to effectively introduce new keypoints whose descriptors do not match with those of the original image, thus concealing the removal forgery; (ii) the perceptual quality of the image following the removal and injection attacks is comparable to the one of the original image.

1. INTRODUCTION

Counterfeiting digital images by means of photo editing tools to alter the original meaning is becoming an immediate and easy practice. Copy-move forgery is the one of the most common ways of manipulating the semantic content of a picture, whereby a portion of the image is copied and pasted once or more times elsewhere into the same image. Image forensics literature offers several examples of detectors for such manipulation [?] and, among them, the most recent and effective ones [?, ?] are those based on Scale Invariant Feature Transform (SIFT) [?]. The capability of SIFT to discover correspondences between similar visual content, in fact, allows the forensic analysis to detect even very accurate and realistic copy-move forgeries. Expectedly, a methodology so powerful has drawn the interest of *counter-forensic* research, where with the term *counter-forensics* the study of methods to counter-attack forensic techniques by concealing manipulations traces is to be intended [?]. The actual reliability of such algorithms can only be estimated by considering what an attacker can try to do to invalidate such techniques. Furthermore, since SIFT is a powerful instrument to recognize and retrieve object, an analysis on SIFT security becomes very important also in the case of Content Based Image Retrieval (CBIR) [?] systems in order to assess if an attacker is able or not to succeed in deluding the image recognition process. The first work in this sense is the one by Hsu et al. [?], in which first the impact of simple attacks is analyzed and then a method to strengthen SIFT features (or keypoints) is proposed. Following this work, Do et al. [?, ?, ?] focused on a SIFT-based Content Based Image Retrieval scenario and devised a number of interesting attacks. The aim of the previous works is to modify the SIFT feature descriptor of a keypoint but they are not interested in the complete removal of the keypoints. A pioneer work on this has been presented in [?] where an attack based on local warping techniques derived from image watermarking was proposed. All these studies have demonstrated that devising methods to attack SIFT feature is not a trivial task. SIFT features are not only robust against several non-malicious processing but also against tampering attempts. Most attacks, in fact, though succeeding in erasing keypoints, pay a high cost in terms of visual quality degradation. Given that, anyway, there is another basic issue to be taken into account when performing keypoint deletion that an image that does not contain SIFT keypoints (or very few of them) is suspicious by itself: such absence, especially in textured areas, could be taken as a clue of tampering, thus leading to a detector whose implementation is very straightforward. Therefore, a smarter attack could greatly benefit from an additional module introducing plausible fake keypoints which could trigger false positives during the SIFT match detec-

*Corresponding author

tion. Reinserted keypoints should ideally appear in a neighborhood of the original spatial locations, but, at the same time, their SIFT descriptors should be as far as possible from the original one in the SIFT space. In addition to that, injection should achieve a number of inserted keypoints as high as possible and, also, a spatial distribution compliant with the underlying image content (a huge number of thickened keypoints could be questionable as well).This topic is crucial in a copy-move forgery detection scenario where portions of an image are to be considered. Such a main aspect is investigated in this paper by analyzing different algorithms to reinsert keypoints in a keypoint-cleaned image while still avoiding matching in the SIFT domain. The fundamental idea of the paper is to highlight the security issue of the need of keypoint injection after a previous removal, to provide some instruments to perform this action and, finally, to present an analysis on some initial results.

The paper is organized as follows: Section **??** introduces the procedure devoted to keypoint removal. Section **??** gives a glance of our idea about keypoints injection in a cleaned image. Section **??** presents experimental results to prove the effectiveness of the proposed method. Section **??** concludes the paper.

2. KEYPOINT REMOVAL

This Section briefly describes the method [**?**] we used to remove SIFT keypoints in a target image by combining different attacks presented in [**?**] and [**?**]; such a method is based on keypoint classification and on an iterative procedure. The idea to adopt a classification of the keypoint typology permits to use an ad-hoc attack for each kind of them, thus maximizing performances. Classification is basically done by resorting to a histogram description of a squared neighborhood around every keypoint; on the basis of the histogram shape three classes are defined: unimodal, bimodal and multimodal. Furthermore, it has been demonstrated [**?**] that sometimes an attack may introduce new keypoints in its attempt to delete those already present. In such cases, a single iteration of the attack is not enough, since there is the need to deal also with the newly introduced keypoints. For this reason, we arranged our attacks into an iterative procedure. For the first part (usually one half) of the iterations we attack all the keypoints with the *Smoothing* attack and for the second part, we alter the keypoints by means of *Collage* and the RMD (Removal with Minimum Distortion) [**?**] attacks according to the previous keypoint classification. The classification-based attack terminates after a certain number of iterations or when the desired percentage of deleted keypoints is reached (ideally 100%). The *Smoothing* attack reduces the population of keypoints without a significant loss of quality. The keypoints that survive to this first round of the attack are somehow "harder" to remove and require more powerful countermeasures (i.e. *Collage* and *RMD*). In the following, we briefly review each attack taken in account.

The first attack is the *Smoothing Attack*. A light Gaussian smoothing flattens the pixel values of an image in such a way that its potential keypoints at the level of DoG are reduced. The strength of the attack can be controlled with the parameters (h, σ), i.e. the size and the standard deviation of the Gaussian kernel. In our experiments we have found out that $h = 3$ and $\sigma = 0.7$ represent a good compromise between the removal rate and the overall visual quality after the attack. This attack has also been used in [**?**].

The second attack is the *Collage Attack*, which is a variant of the attack used in [**?**]. It consists on the substitution of the original patch with another patch of the same size. The new patch should not contain a keypoint and needs to be as similar as possible to the original one according to some criteria of similarity. To implement the collage attack we created a database of about 120000 "keypoint-free" patches extracted from a data set of 80 images characterized by very heterogeneous visual contents. We chose to measure the similarity by means of the histogram intersection distance, which has been widely used in the past in image retrieval applications [**?**]. Let now $patch_{orig}$ and $patch_{min}$ be respectively the original patch and the most similar counterpart stored in the database (i.e. the patch whose histogram is at minimum d_{int}); to avoid visible artifacts along the borders, we do not reinsert $patch_{min}$ directly into the original image. Instead, we reinsert the following linear combination:

$$patch_{new} = W \cdot patch_{orig} + (1 - W) \cdot patch_{min} \quad (1)$$

where W is an empirical 8×8 weighting matrix whose elements $w_{i,j} \in [0, 1]$ are set to 1 along the patch borders and progressively decrease to 0 near the center.

The third attack is the *RMD* attack proposed by Do et al. in [**?**]. The idea behind this technique is to calculate 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 computed at the DoG level, thus invalidating the check performed by the SIFT algorithm on all potential keypoints. Moreover, it is requested that the coefficients locally introduce the minimum visual distortion and, differently from the original version of the algorithm, we used the same weighting window of Eq. (**??**) to replace the original neighborhoods with the new patch.

3. KEYPOINT INJECTION

In this Section, the injection of fake keypoints into the cleaned image, obtained through the procedure described in Section **??**, is investigated. However, before discussing this issue in depth, a schematization of the whole attack procedure (keypoint removal and injection) is shown in Figure **??** for sake of clarity.
At the beginning an original gray-scale image I (or a region within it) is fed to the system, which starts by detecting the SIFT keypoints. Then, for each keypoint, the corresponding 8×8 patch is manipulated by means of the attack procedure described in Section **??**. Finally, the manipulated patches are inserted back into the image in their original positions to achieve keypoint removal and the cleaned image I_c is obtained.
The second step of the procedure is devoted to the injection of fake keypoints into the cleaned image. We have tested several different injection algorithms (e.g. contrast enhancement, sharpening and so on) to introduce new keypoints into the cleaned image. The most specific and promising ones, taken into account in the experimental tests presented in Section **??**, will be described in subsection **??**. At this stage, the image is processed full-frame, unlike in the previous stage, so this fact has a negative impact on the visual quality of the entire image (e.g. flat areas which did not contained keypoints originally). For this reason, in a 8×8 neighborhood of each keypoint we mix this image, let us call it pre-injected image, with the original one I, in a way that is similar to what happened in Eq. (**??**). The obtained patches are then substituted onto the cleaned image I_c producing the final injected image I_{inj} which now shows a better visual quality.

In the third step, it is necessary for the attacker to check how many injected keypoints in the image are really valid. An injected keypoint is deemed as valid when, first of all, is located in a textured area and is spatially distributed with respect to the others and, above all, has a SIFT descriptor which is sufficiently different from its original homologue not to evidence a match. It is worth to point

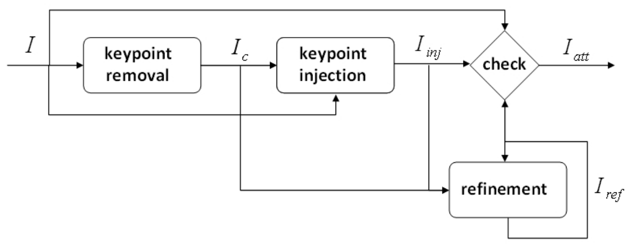

Figure 1: Schematization of the proposed framework.

out that it is not so crucial to check that the new injected keypoint is or is not in the same spatial position (x,y coordinates) with respect to the original one. So, on this basis, we perform a matching detection between the original image I and the injected one I_{inj}. Ideally, it would be desirable not to obtain matches between the two images, though having in the injected image I_{inj} a plausible amount of well distributed keypoints: the final image I_{att} is obtained. It has also been foreseen, a possible refinement phase (at the moment only one loop is considered) where a selection of keypoints is made by taking the valid ones and discarding those presenting a correct match with their homologue (sometimes wrong welcomed matches are obtained, see Section **??** for details). For each discarded keypoint, a corresponding patch (16×16 pixels which is the computational window of the SIFT descriptor) of the cleaned image I_c is selected and inserted back into the image I_{inj} refining the injection procedure. The background idea is to primarily avoid a SIFT match at the expense of the loss of an injected keypoint.

3.1 Injection Algorithms

In this subsection, we briefly describe the techniques that we have employed to inject fake keypoints into an image. Their choice is justified by the following idea: we observed that smoothing techniques are quite effective in removing SIFT keypoints by lowering the image details. Therefore, we hypothesized that enhancement techniques, which exalt image details, may conversely introduce new keypoints. However, full-frame enhancement techniques, such as sharpening or global contrast enhancement, proved to be visually unsatisfactory, since they do not take into account local properties of the image. As a consequence, we needed to resort to more complex methods; hereafter four methods that have demonstrated superior performances and that have been selected for being presented within experimental result section of the paper are debated.

3.1.1 Contrast limited adaptive histogram equalization

Global contrast enhancement (GCE) techniques assume that the distribution of grayscale pixel values is uniform over all the areas of an image. When this assumption does not hold, GCE performance are poor and the resulting image visually unpleasant. The *CLAHE* (Contrast Limited Adaptive Histogram Equalization) [**?**] tackles this problem in two ways: first, it adapts to the local properties of the regions of an image and, secondly, it limits the contrast differences across them. In a nutshell, the algorithm proceeds as follows (see [**?**] for details). First the image I is divided into a specified number of non-overlapping regions (tiles) and the histogram of each region is computed. Then, a clipping limit β for the contrast enhancement is obtained by means of the following Eq. (**??**):

$$\beta = \frac{MN}{L}\Big(1 + \frac{\alpha}{100}(s_{max} - 1)\Big) \qquad (2)$$

where: $[M, N]$ are the size of the grayscale image, $L = [0, 255]$ the histogram bins, $\alpha \geq 0$ is the *clipping factor* and s_{max} is the slope of the transfer function mapping the contrast from its input value to its output value; if $s_{max} = 1$ then no enhancement is performed, while larger values (usually up to 4) will result into more visible enhancements. Next, each region's histogram is clipped in such a way that its height is limited by β. At this point, it is necessary to remap the clipped values to the entire intensity range (i.e. to re-normalize the histogram of the processed image to its original area). This task can be done in several ways, the most common of which consists on redistributing the clipped pixels uniformly in all the bins of the histogram of the whole image.

3.1.2 Brightness preserving dynamic fuzzy histogram equalization

The *BPDFHE* (Brightness Preserving Dynamic Fuzzy Histogram Equalization) [**?**] is a method to enhance the contrast of an image while preserving its mean brightness, and thus the perceived subjective quality of the image. Similarly to other contrast enhancement techniques, the BPDFHE proposes to divide the image histogram into segments, which are then independently equalized. The partitioning, however, is not performed on the normal histogram, but rather on its fuzzy counterpart, whereby a pixel may belong to some degree to more than one of the bins, in accordance with a fuzzy membership function. Such histogram, in facts, is smoother, with no missing levels or abrupt fluctuations, thus allowing a more accurate segmentation. The membership function can be designed in different fashions, in our experiments *triangular* and *gaussian* have been considered.

The algorithm proceeds as follows (see [**?**] for details): (i) the fuzzy histogram $\tilde{H}(k)$, $k = [0, 255]$ is computed by assigning to each bin k the number of pixels whose value is "around k" (according with the chosen membership function); (ii) the local maxima $\{m_1, m_2, \ldots, m_n\}$ are computed and used to define histogram's segments $S = \{[\tilde{H}_{min}, m_1 - 1], [m_1, m_2 - 1], \ldots, [m_n, \tilde{H}_{max}]\}$, where $[\tilde{H}_{min}, \tilde{H}_{max}]$ is the range of $\tilde{H}$; (iii) each segment is equalized by means of a technique depending on the number of pixels belonging to the partition; (iv) in order to cope with the alterations that BPDFHE may have introduced, the resulting image's brightness is finally normalized to match the original brightness.

3.1.3 Anisotropic diffusion

The *2D-Anisotropic Diffusion* (2D-AD) is a method to enhance images by preserving the perceptual quality of semantically relevant parts (i.e. straight lines, edges, geometric shapes) [**?**]. In principle, it is a generalization of the scale-space transform, where an image I is iteratively convolved with a nonlinear smoothing filter which adapts to the local content to generate progressively more blurred versions of I. In this paper, we resort to the works of Weickert [**?**, **?**], to which we refer for theoretical details.

The filter model used for anisotropic diffusion is derived by well-known operators used to extract image details. Let $I_\sigma = I * G_\sigma$ be the convolution of an image I with a Gaussian kernel ($\sigma > 0$); then, the gradient ∇I_σ can be employed to highlight structures such as the edges of I, unless they are parallel. In this case, a more accurate method is required. Let $J(\nabla I_\sigma) = \nabla I_\sigma \nabla I_\sigma^T$ be the Hessian of ∇I_σ; then, $J_\rho(\nabla I_\sigma) = J(\nabla I_\sigma) * G_\rho$, that is the convolution with a Gaussian kernel ($\rho > \sigma$), is called *tensor operator*, and it can be used to effectively highlight parallel, flow-like or T-shaped structures [?]. The eigenvectors $\{w_1, w_2\}$ of J_ρ give indications on local orientations, and the corresponding eigenvalues (μ_1, μ_2) on the local contrast along these directions. The *diffusion tensor* D, that permits to perform the anisotropic diffusion, is defined by means of the eigenvectors of J_ρ and the eigenvalues of Eq. (**??**-**??**).

$$\lambda_1 = c_1 \tag{3}$$

$$\lambda_2 = \begin{cases} c_1 & \text{if } \mu_1 = \mu_2 \\ c_1 + (1 - c_1)\exp\left(\frac{-c_2}{(\mu_1 - \mu_2)^2}\right) & \text{otherwise} \end{cases} \tag{4}$$

where $c_1 \in (0, 1)$ and $c_2 > 0$. By resorting to D, it is possible to efficiently compute blurred versions of $I(x, t)$ as numerical solutions of Eq. (**??**), where $t \geq 0$ is called *diffusion time*.

$$\frac{\partial I}{\partial t} = \nabla \cdot (D \nabla I) \tag{5}$$

In practice, the algorithm proceeds as follows: given $I = I(x, 0)$, first $J_\rho(\nabla I_\sigma)$ is computed and D is derived (Eq. (**??**-**??**)); then, $I(x, 1)$ is obtained (Eq. (**??**)). Starting from $I(x, 1)$, the process is repeated until a specified number of iterations have been reached (i.e. $t \leq t_{max}$). Consequently, the final processed image corresponds to $I(x, t_{max})$. The tensor D ensures that each iteration will, at the same time, preserve linear structures and smooth the image along them. The model can be further refined in such a way that orientations are invariant to rotation [?].

4. EXPERIMENTAL RESULTS

In this Section, experimental tests carried out to check the performances of the proposed procedure for keypoint removal and injection are presented. A dataset of 35 digital images has been created by randomly drawing from UCID database [?] with size ranging from 300×400 to 600×800 pixels and with different visual content: landscapes, animals and faces. We will evaluate the performance of the proposed method both from the point of view of number of injected keypoints and of number of matches obtained after injection. In the following tests the keypoints have been computed by means of `VLFeat`, the Vedaldi and Fulkerson's implementation of SIFT [?] (DoG peak and edge thresholds set to 4 and 10). The threshold for keypoint matching is fixed to 0.6, as suggested by Lowe in [?]. We set a target keypoint removal percentage of 100% (perfect removal) and a maximum number of allowed iterations (i.e. $max\_iter = 40$). A higher number of iterations would have an unacceptably negative impact on quality. We continued with the iterations until we reached the desired removal percentage or the 40-th iteration. Then, after the keypoint removal, we applied the injection module. Results refer to four injection tools (*CLAHE*, *BPDFHE-Tri/Gauss* and *2D-AD*) presented in subsection **??**; such methods have been used by adopting a parameter setting according to the values indicated in their reference papers. Section **??** is organized into two subsections: subsection **??** presents a quantitative analysis regarding the results achieved by the injection operation, while subsection **??** discusses some specific cases in detail, paying attention to the issue of *correct* generated matches. With this meaning *correct*, all the SIFT matches that link two keypoints, respectively belonging to two images, located in the same spatial position or, at most, within a 8×8 neighborhood are intended.

4.1 Effectiveness of keypoint removal-injection

In this subsection an analysis on the effectiveness of the procedure for keypoint removal-injection is presented. In Figure **??** (left column), the number of keypoints detected for each image, belonging to the selected dataset, is plotted. In particular, we have two reference trends that are common to both graphs (top-left and bottom-left): the number of keypoints existing in the original image (*Original*) which represents the upper bound and the ones remained in the cleaned image after removal (*Clean*) which, on the contrary, can be considered as the lower bound. Between these two, the trends of keypoints after injection performed with the four described tools are plotted: *BPDFHE triangular* and *gaussian* in the top-left graph, *CLAHE* and *2D-AD* in the bottom-left one. It can easily be noticed how keypoint removal drastically reduces the number of keypoints which is globally close to the zero level though, in some cases, a quite consistent amount of keypoints survives, like for images 1, 6 and 20. After that, injection substantially succeeds in raising the number of keypoints per image towards the original quantities: all the four tools show similar behaviors (it is out of the scope to determine the best performing tool), with some borderline situations: sometimes no keypoints are injected (e.g. images 29, 30 and 31 top-left), sometimes more than the original (e.g. image 18 bottom-left).

On the other side, in Figure **??** (right column, again tools *BPDFHE triangular* and *gaussian* in the top-right, *CLAHE* and *2D-AD* in the bottom-right), histograms of the deviations ($diff$) between the number of SIFT matches obtained between the original image and the injected one, and the original image and the cleaned one are plotted. It can be observed that histograms are globally located around zero that means that, though keypoint injection has been carried out, a low amount of additional matches is actually appeared with respect to the case of the cleaned image. Negative cases indicate that an inferior number of matches is present after the injection step.

4.2 An in-depth analysis

In this subsection, we have extracted two images ($I15$ and $I35$) from the previous dataset of 35 images, trying to provide a different analysis point of view of the proposed approach; the tool *BPDFHE-Triangular* is taken into account for both. In Figure **??**, the case of image $I15$, named *Dwarf*, is pictured; in top row keypoints are visualized respectively, from left to right, for original, cleaned and injected images. It can be seen that the cleaned image (top-center) contains only three keypoints while the injected one presents a higher amount of spatially distributed green circles. In particular, now keypoints appear on the nose of the dwarf, as primarily was, and, interestingly, on the top of the hood where there were not in origin. In the bottom row of Figure **??**, SIFT matches are highlighted: between the original image and the cleaned one (bottom-left), between the original and the injected (bottom-right). In this favorable circumstance, after keypoint removal we obtained that three correct matches and two wrong (not horizontal lines) are left, but, after injection, we got again three correct matches, though one is different (keypoint on the dwarf nose), and one wrong. This means that injection operation does not determine an unwanted improvement in image matching detection.

In Figure **??**, in the same presentation structure as before, the case of image $I35$, named *Bell Tower*, is pictured. Similar results are obtained as previously, but two aspects are interesting in this case. Concerning the issue of keypoint extraction: cleaned image (Figure **??** top-center) does not contain any keypoint, that is quite suspicious per se being the image content very textured, as clearly evidenced by the original distribution of keypoints (Figure **??** top-left); instead injected image is more likely: the absence of keypoints over

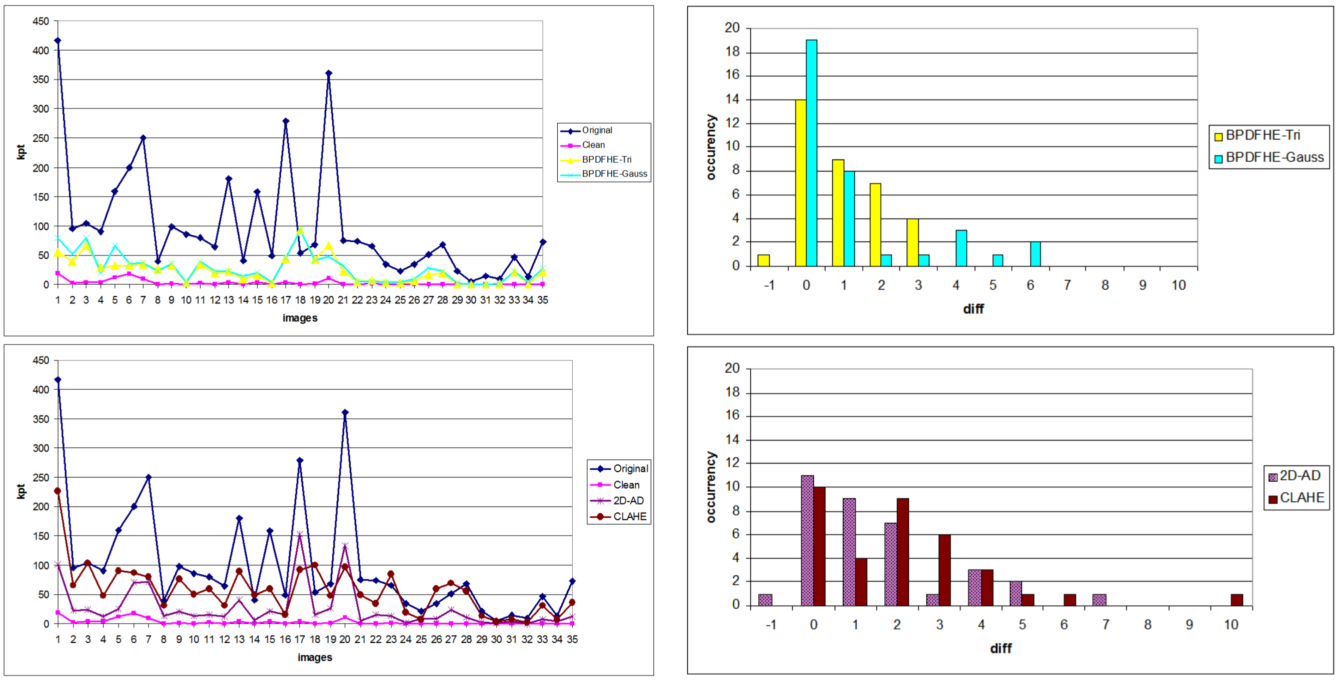

Figure 2: Graphs for keypoint removal-injection procedure. *Left column*, keypoints per image and *Right column*, variation of SIFT matches per image.

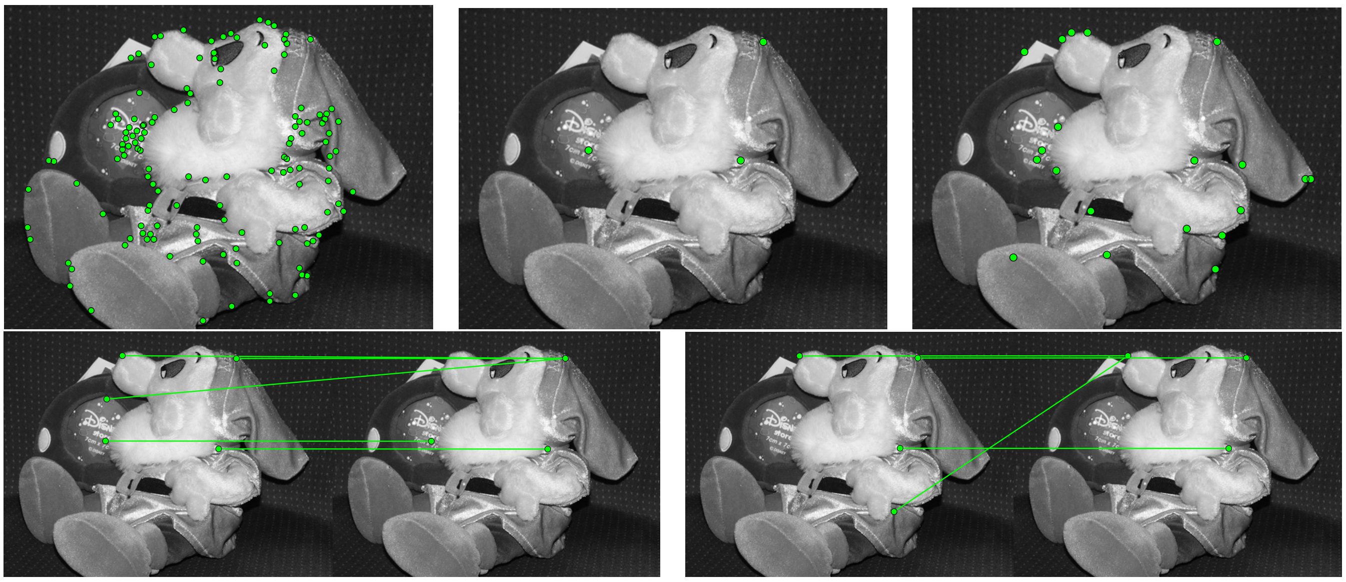

Figure 3: Image $I15$ (*Dwarf*). On the top row, keypoints are shown: original image (left); cleaned (center); injected (right). On the bottom row, SIFT matches: original vs cleaned (left); original vs injected (right).

the clouds, which is an almost flat area, is not so strange. Concerning the issue of SIFT matches: obviously, the cleaned image, not containing any keypoint, does not produce any match with the original at all (Figure **??** bottom-left), but the injected one, which in turn has 4 matches (Figure **??** bottom-right), does not present any correct. In fact, looking carefully at the green match line in the highest part of Figure **??** (bottom-right), it can visually be appreciated that also such line is not horizontal and, as explained at the beginning of Section **??**, this means that the fake keypoint falls out of a 8×8 neighborhood with respect to the location of its possible homologue. Such an aspect could become crucial when an operation of estimate of the geometric transformation existing between the two images is carried out based on these matches; this might be necessary, for instance, for image registration or for region duplication localization in forgery detection in a forensic scenario. In such a circumstance, this would yield to a wrong computation and consequently to a misleading result.

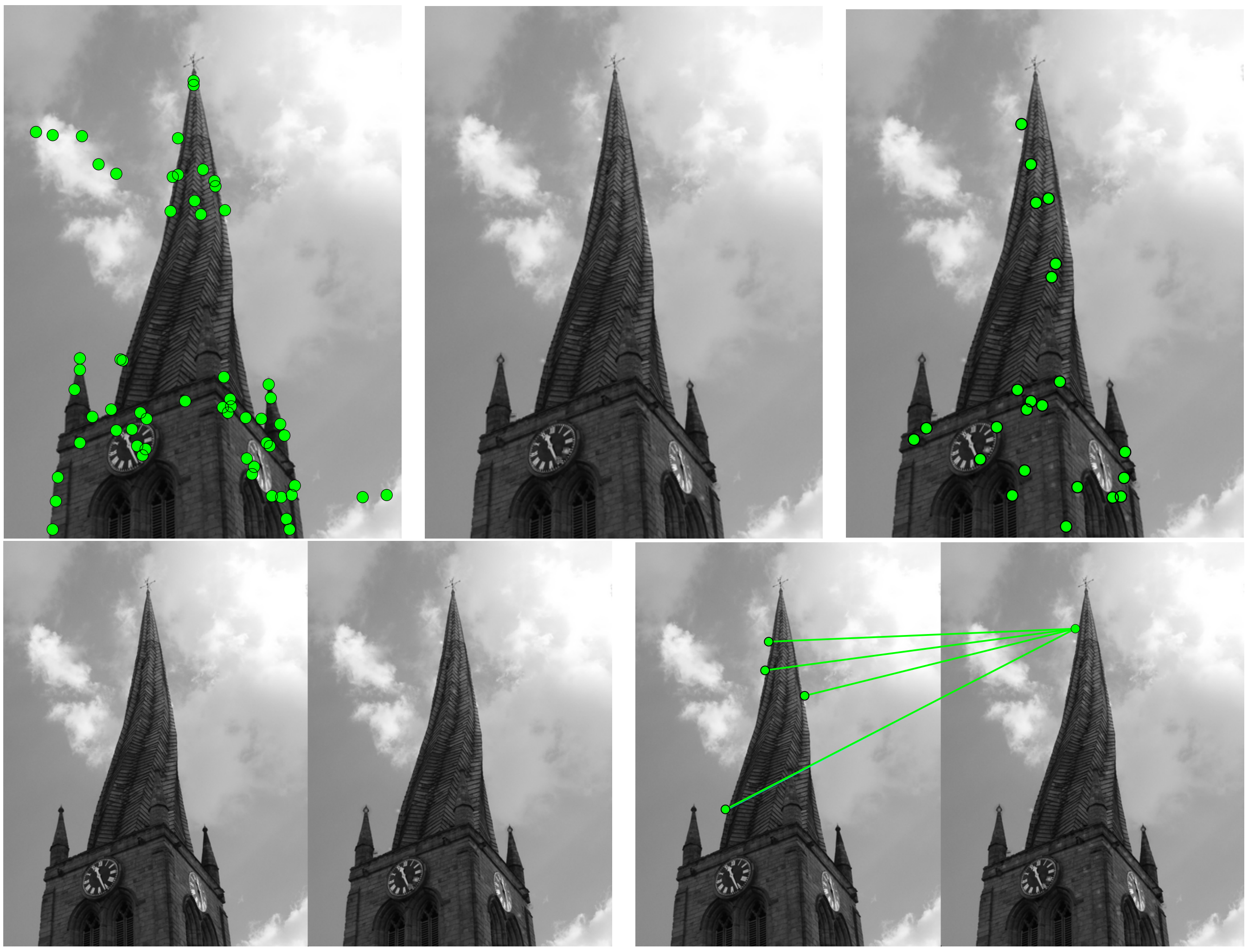
Figure 4: Image $I35$ (*Bell Tower*). On the top row, keypoints are shown: original image (left); cleaned (center); injected (right). On the bottom row, SIFT matches: original vs cleaned (left); original vs injected (right).

Hereafter, detailed values, both for keypoints and for SIFT matches, for the two previous images, are provided in Table **??**, to allow a numerical analysis of what visually proposed so far. In the left column of Table **??** with term *Correct SIFT matches* all the right matches, as explained previously, between the original image and the cleaned (injected) one are intended again. As already stated throughout the paper, here it can be observed that image quality (see PSNR values at the bottom of Table **??**) is not worsened after injection phase in comparison with what achieved after keypoint removal only; visual quality with respect to the original image is satisfactorily preserved too. Indicatively, such a behavior has globally been registered for all the images of the selected dataset.

5. CONCLUSIONS AND FUTURE WORK

In this paper, the basic issue to inject fake SIFT keypoints in a previously cleaned image has been underlined and investigated. This has been moved from the consideration that a complete keypoint-free image could be per se a clue of counterfeit. Furthermore, a procedure to firstly remove and then reinsert keypoints has been presented and a set of possible injection tools has been analyzed. Experimental results are encouraging and already show that injection is feasible without causing a successive detection at SIFT-matches level. Visual quality is still preserved both with respect to the original image and, particularly, in comparison with the image quality achieved after keypoint removal only. Future works will be dedicated to the research of more effective and ad-hoc injection tools and to evaluate the whole procedure against a wider image dataset.

Acknowledgment

This work was partially supported by the REWIND Project, funded by the Future and Emerging Technologies (FET) programme within the 7FP of the EC under grant 268478.

Table 1: Performance of keypoint removal-injection attack on two sample images: $I9$(*Dwarf*) and $I35$(*Bell Tower*).

Description	**I9 (*Dwarf*)**	**I35 (*Bell Tower*)**
Original image keypoints	158	73
Cleaned image keypoints	3	0
Injected image keypoints	20	27
SIFT matches Original-vs-Cleaned	5	0
Correct SIFT matches Original-vs-Cleaned	3	0
SIFT matches Original-vs-Injected	4	5
Correct SIFT matches Original-vs-Injected	3	0
PSNR Original-Cleaned	39.12 dB	42.55 dB
PSNR Original-Injected	39.12 dB	42.02 dB