Automatic Image Splicing Detection Based on Noise Density Analysis in Raw Images

Thibault Julliand[1(✉)], Vincent Nozick[2], and Hugues Talbot[1]

[1] Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, 93162 Noisy-le-Grand, France
{thibault.julliand,hugues.talbot}@esiee.fr
[2] Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, 77454 Marne-la-Vallée, France
vincent.nozick@u-pem.fr

Abstract. Image splicing is a common manipulation which consists in copying part of an image in a second image. In this paper, we exploit the variation in noise characteristics in spliced images, caused by the difference in camera and lighting conditions during the image acquisition. The proposed method automatically gives a probability of alteration for any area of the image, using a local analysis of noise density. We consider both Gaussian and Poisson noise components to modelize the noise in the image. The efficiency and robustness of our method is demonstrated on a large set of images generated with an automated splicing.

Keywords: Image forgery · Noise · Raw image

1 Introduction

The number of digital images has hugely increased over the last decades. Their sources have diversified with the arrival of smartphones and tablets, and they form the majority of the pictures we see nowadays. The number of falsified images has increased accordingly, both in number and sophistication, with tools more and more efficient to do so. Farid [4] explains the rise of digital forensics to combat this trend.

In this paper, we show a new method to detect a common falsification called splicing. Splicing is one of the more common forms of image alteration. It consists in inserting part of an image in a second, different image. Our method is based on exploiting the noise density of an image. We focus here on raw images, with the assumption that the noise is unaltered. Although raw images are harder to tamper with than more common image formats, it is still possible, especially with the DNG open file format [9].

1.1 Image Splicing Detection

Various ways to detect splicing already exist, however most of them cannot be used on raw images: Farid bases his on JPEG ghosting [5] or on an analysis of

J. Blanc-Talon et al. (Eds.): ACIVS 2016, LNCS 10016, pp. 126–134, 2016.
DOI: 10.1007/978-3-319-48680-2_12

Color Filter Array perturbations with Popescu [18]. Lin et al. [11], He et al. [8], and Popescu et al. [17] exploit the quantization in JPEG images. Some other methods are either very high-level or have hard to meet prerequisites. Lukas et al. [12] present a method based on the camera fingerprint, but this method requires some unaltered images taken by the camera, or access to the camera itself. Machine learning is also a possibility, such as presented by Bayram et al. [2] or Fu et al. [7]: a classifier of image features learned from training sets of authentic and forged images can be used to detect spliced regions in an image.

1.2 Splicing Detection from Image Noise

Noise is a perturbation that can be found in all images captured by a digital sensor. Although this noise may be reduced by the camera internal pipeline or with post-processing, the noise in a single image will have the same parameters throughout the image. These parameters will vary according to the camera model and the light exposition during the image capture. Thus, observing a variation in noise parameters in a specific zone of an image will often be a strong indicator of falsification. Consequently, some methods use image noise to detect splicing regions. In most noise estimation methods used in digital forensics, noise is simplified as Gaussian. This approach ignores the Poisson component in raw images, which tends to be dominant in high-intensity zones.

Mahdian and Saic [14] use a block-based inconsistency detection relying on the homogeneity of the standard deviation of the Gaussian noise. This approach, however, relies on the assumption that the noise standard deviation is homogeneous over an image. This is not always true, especially in raw images where the noise also includes a Poisson component. Pan et al. [15,16] also propose a block-based approach. Their method uses an analysis of kurtosis values in an image. Although this method is very efficient at detecting noise inconsistencies, it less efficient on images including textures and low noise values, increasing the difficulty of splicing detection in natural images. Popescu and Farid [17] also use noise estimation to detect splicing with excellent results, but their method needs preliminary informations about the noise of the original image, which can not be done in the case of a blind analysis. Finally, Julliand et al. [10] offer an approach which takes into account the Poisson component, but their method lacks precision in the localisation of the altered region.

Some methods are a combination of various approach. For example, Mahdian and Saic [13] combine resampling detection and noise analysis to highlight suspicious areas.

2 Noise Density Contribution Tables

2.1 Principles and Definitions

Noise in digital images can come from a wide variety of sources. In a raw image, noise follows a Poisson-Gauss probability distribution, with the standard deviation varying with the intensity of each pixel. A noise density table is the representation of this probability distribution. For each pixel, we consider its denoised

value v_d and its noised value v_n. To each pixel of the image corresponds a value pair (v_d, v_n), which are accumulated in the table. This way, the table can be seen as a 2D histogram, as depicted in Fig. 1. The exact function defining the Poisson-Gauss probability density table is shown in Eq. 1, where σ is the standard deviation of the Gaussian portion of the function and α a scaling parameter applied to the Poisson portion:

$$f(v_d, v_n) = \frac{\alpha}{\sigma\sqrt{2\pi}} \sum_{x=0}^{\infty} \frac{(\alpha v_d)^{\alpha x} e^{-\alpha v_d}}{(\alpha x)!} \exp\left(\frac{-(v_n - x)^2}{2\sigma^2}\right) \tag{1}$$

In practice, the value of the table at any point (i, j) is the number of value pairs where $(v_d, v_n) = (i, j)$. For numerical purposes, we normalize the table on each row (denoised values) to offset potential intensity imbalances in the image. Indeed, the table of an image with a high proportion of high (or low) intensity pixels would have very high values in the corresponding areas. This would reduce the usability of the table. The normalization suppresses this problem, as shown in Fig. 1.

A cross-section of the table along a single denoised value follows a Poisson-Gauss probability distribution (see dark line in Fig. 1). However, in the case of a spliced image, the noise density will be the sum of two different noise probability functions: one for the original image, and one for the spliced element (Fig. 2). The objective of our method is to differentiate these two contributions, and to identify the parts of the image that participate in each contribution.

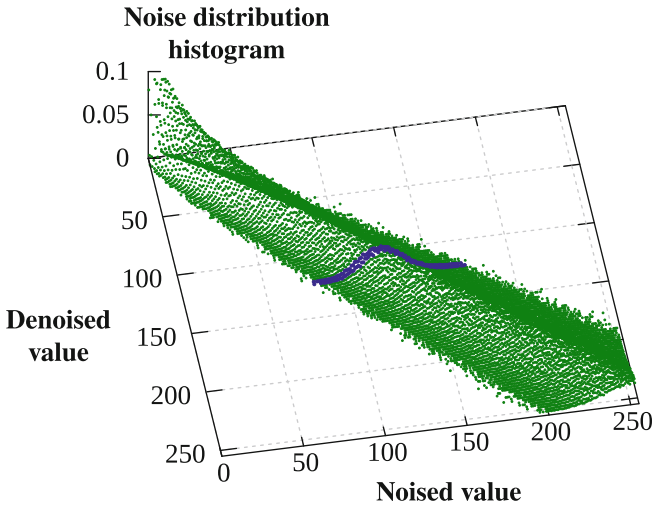


Fig. 1. Poisson-Gauss density map. The dark line is a cross-section along a single denoised value.

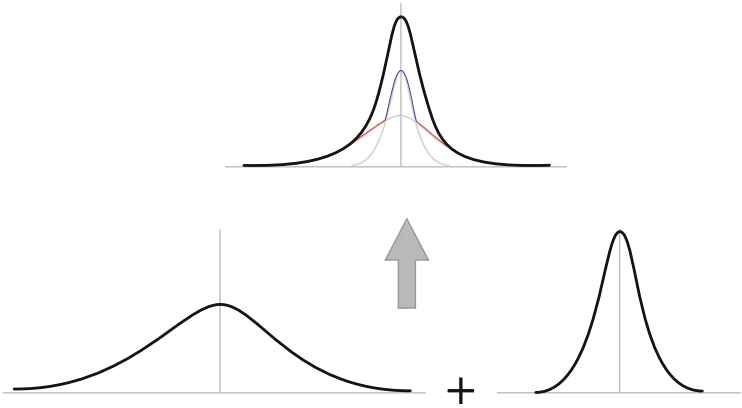


Fig. 2. Density map is an addition of two curves.

A naive approach would be to try to fit a model, such as the one in Eq. 1, based on the overall noise characteristics over the noise density table, to try and see which parts can be considered as outliers. However, this proves to be ill adapted for two reasons: first, unless the spliced area represents a significant portion of the original image, the impact on the noise density table will not be noticeable. Second, the normalization process will flatten any major and noticeable difference.

2.2 Noise Density Contribution Table

A noise density contribution table (referred to as "contribution table" from here on) represents the contribution percentage of any subimage of an image to the noise density table of the full image − more specifically, its contribution to each of the (v_d, v_n) value pair presented before. Basically, a contribution table C_{sub} is the noise density table D_{sub} of a subimage divided by the noise density table D_{im} of the whole image. A contribution can never be more than 1, 1 meaning that all the pixels contributing to a pair are included in the subimage. More formally, we get:

$$C_{sub}(v_d, v_n) = \frac{D_{sub}(v_d, v_n)}{D_{im}(v_d, v_n)}, \quad \forall (v_d, v_n)$$

As a consequence of the overall noise being the sum of two different noises, two shapes of contribution tables in a spliced image will appear. This is due to the impact of each type of noise on the global one: as we can see on Fig. 2, each curve will have a zone with higher participation. The first will have higher contributions on the identity axis, and the second higher contributions outside of the identity axis, respectively referred to as $\wedge$ type and $\vee$ type.

2.3 Classification Between ∧ Type and ∨ Type

The next step is to define the subimages and identify the type of their contribution tables. To do so, the image is divided into a high number of square blocks of identical size. Each of these blocks will be considered as a subimage, and will have its own contribution table, see Fig. 3(a) and (b). To identify the type of a contribution table, we locate its M highest contributions (corresponding to the M maxima of the table). According to the location of these maxima, a type will be attributed to the block: if there is a clear majority of them on, or near, the identity axis, it will be a ∧ type. If there is a clear majority outside of this axis, it is a ∨ type (Fig. 3(c)). If none of those conditions are fulfilled, the type remains undefined.

To make the method more robust, a good approach is to increase the number of pixels in the subimages. Indeed, contribution tables are easier to identify when they are built from more pixels. However, increasing the size of our blocks would greatly reduce the spatial precision of our detection. To increase the robustness while keeping the same precision, we create overlapping square cells, each containing a moderate number of blocks. The contribution table of a cell is the sum of the contribution tables of the blocks it contains. This results in contribution tables which are easier to identify, thanks to the higher amounts of pixels used in each cell. The type of each block then corresponds to the type in majority present in the cells containing it. If there is no clear majority, the block type is undefined and it will be changed in the seed expansion phase (see Fig. 4, middle column).

3 Seed Expansion

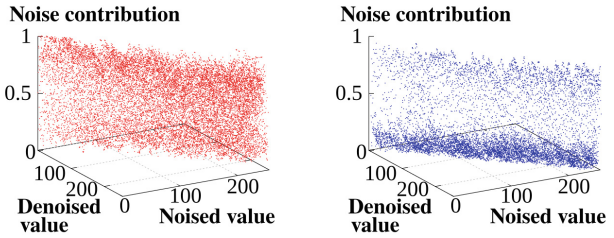
Once every block has been assigned a primary type (be it ∧, ∨, or undefined), we begin the expansion to find which of the two main categories each undefined block is more likely to belong to. This expansion is based on the similarity between blocks and the assumption that two similar blocks will probably belong to the same type ∧ or ∨. The similarity s between two contribution tables $C_1(v_d, v_n)$ and $C_2(v_d, v_n)$ is simply a sum of term by term absolute difference, but only in the rows where both tables have non-zero values:

$$s = \sum_{i \in \mathcal{D}} \sum_{j=0}^{v_n^{max}} |C_1(i,j) - C_2(i,j)|$$

where

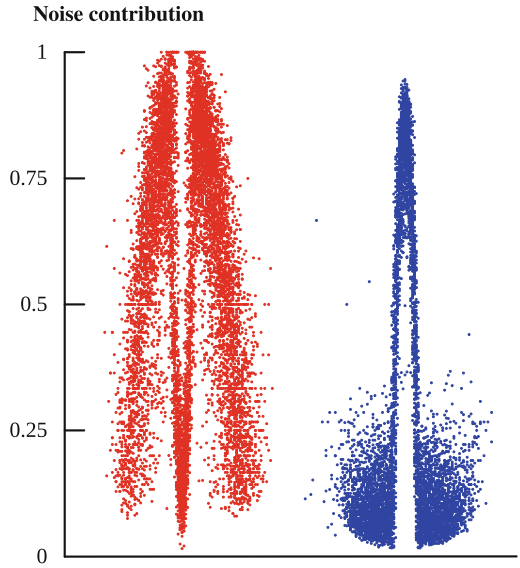
$$\mathcal{D} = \left\{ i \mid \sum_{j=0}^{v_n^{max}} C_1(i,j) \neq 0 \text{ and } \sum_{j=0}^{v_n^{max}} C_2(i,j) \neq 0 \right\}$$

For a higher accuracy, we assign to each undefined block a probability p to belong to the ∧ shape group, and thus a probability $1 - p$ to belong to the ∨ shape group. These probabilities are computed with an iterative scheme with an initial



(a) $\vee$ contribution table

(b) $\wedge$ contribution table



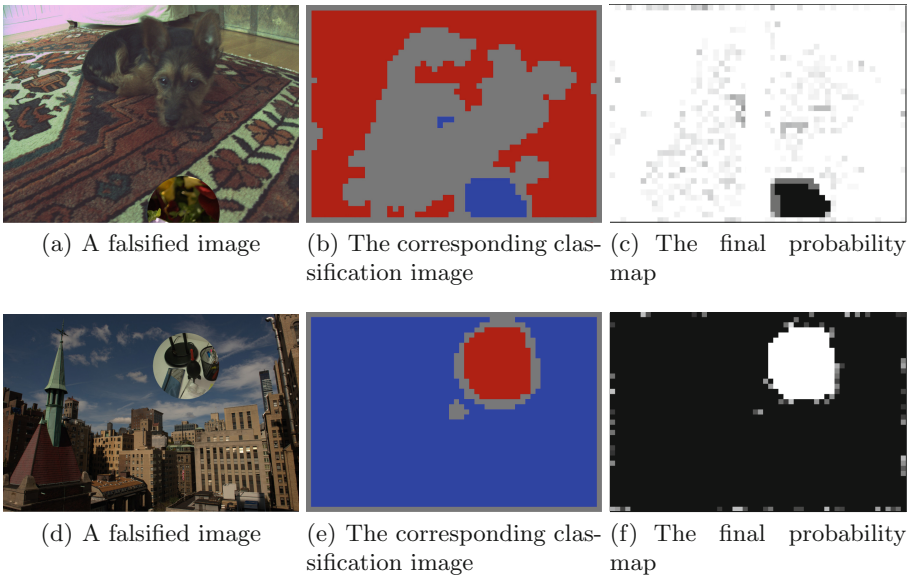
(c) Projections of each contribution table

Fig. 3. The two types of contribution tables. (a) and (b) show their overall appearance, (c) shows their projection on a plane orthogonal to the identity axis.

value set to 0.5. Then, each undefined block probability is iteratively set as the weighted average probability of the N blocks whose contribution table is the most similar to that of the current block, with higher similarities giving a higher weight. At an iteration k, the probability p_b^k of an undefined block b is:

$$p_b^k = \frac{\sum_{n=1}^{N} \frac{p_n^{k-1}}{n}}{\sum_{n=1}^{N} \frac{1}{n}}$$

This process is iterated on all undefined blocks until convergence, or until a set amount of iterations have been reached. Convergence can be reached in two



(a) A falsified image (b) The corresponding classification image (c) The final probability map

(d) A falsified image (e) The corresponding classification image (f) The final probability map

Fig. 4. Left column: spliced images. Middle column: block types after the initial classification. Blue zones are $\wedge$, red zones are $\vee$, and grey zones are undefined. Right column: Final result after the seed expansion. (Color figure online)

different ways: first, if a block probability is high enough, the block is considered as being fully in said category. Computation stops when each block of the image is fully in one category or the other (meaning no undefined block left). Second, when the total probability changes δ in the image from one iteration to another is under a value ϵ set by the user. δ is defined as:

$$\delta = \sum_{b \in \mathcal{B}} |p_b^k - p_b^{k-1}|$$

where $\mathcal{B}$ is the ensemble containing all blocks of the image. The result can be seen in Fig. 4, right column.

4 Multichannel

In order to improve our results, we apply the whole process - denoising, noise density table, contribution tables, classification, and expansion - to each channel of the input image (R, G_1, B, G_2). Although the grey world assumption [6] can be applied on a multi-channel image, each channel can contain drastically different information − a clear sky, for example, will appear with a much higher intensity on the blue channel. As each channel can give a more precise information on various parts of the image, the multi-channel approach grants a higher precision and more robustness in the final result. To do so, the probability map of each

channel are averaged, with each map having an equal weight. As we work with raw images, we have 4 channels to work with − including two green channels. No bias towards one particular channel was observed in our experiments.

5 Implementation and Results

The raw images are loaded using LibRaw [1]. The denoising process is performed by BM3D using the Matlab code provided by [3]. Not taking the denoising time into account, which we have little control over, the multi-channel version of the code runs in around a minute for a 2000×2000 pixels image on consumer-grade hardware.

The choice of the code handling the denoising part is a crucial part of the method: indeed, our method is extremely dependent on the quality of the denoising. Although the procedure we used [3] is state-of-the-art for Poisson-Gauss denoising, it tends to produce relatively poor results on dark textured areas. Even though our method has no theoretical weakness on such areas, due to this, our output quality drops similarly on images containing this kind of elements.

For our experiments, we used a base of 290 spliced images and 27 authentic images. Those images come from a wide variety of cameras: Canon (2 models), Leica (4 models), Nikon (1 model), Panasonic (9 models), Pentax (3 models), and Sony (3 models). The results are exposed in Table 1. However, those results do not take into account images containing dark textured areas (respectively 73 spliced and 7 authentic). If those images are considered, the splicing localization rate drops to 51.7 %, and the authentic detection rate to 58 %. The splicing detection rate remains at 100 %.

Our method's effectiveness is likely to increase in accordance to the efficacy of upcoming denoising methods.

Table 1. The detection rate on spliced and unspliced images.

Image type	falsified/authentic correctly identified	Splicing localization correct
Spliced	100 %	68.6 %
Authentic	86 %	na

6 Conclusion

We present a new method to automatically detect splicing in raw images. This method is based on the discrimination of contributions in the noise density of the image, when the image contains a spliced element. By looking at the locations where the contribution to the noise is different, we show that it is possible to pinpoint a spliced area in an image. The robustness of the approach is increased by replicating it over all the channels of the image. Further research will aim to adapt this method for JPEG images.

References

1. LibRaw-0.17: Image decoder library (2015). www.libraw.org
2. Bayram, S., Avcibas, I., Sankur, B., Memon, N.D.: Image manipulation detection. Electron. Imaging **15**(4), 1–17 (2006)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3D transform-domain collaborative filtering. IEEE Trans. Image Process. **16**(8), 2080–2095 (2007)
4. Farid, H.: A survey of image forgery detection. IEEE Signal Process. Mag. **26**(2), 16–25 (2009)
5. Farid, H.: Exposing digital forgeries from JPEG ghosts. IEEE Trans. Inf. Forensics Secur. **4**(1), 154–160 (2009)
6. Finlayson, G., Shiele, B., Crowley, J.: Comprehensive colour normalization. In: Proceedings European Conference on Computer Vison, vol. I, pp. 475–490 (1998)
7. Fu, D., Shi, Y.Q., Su, W.: Image splicing detection using 2D phase congruency and statistical moments of characteristic function. In: Proceedings of SPIE Security, Steganography, and Watermarking of Multimedia Contents IX (2007)
8. He, J., Lin, Z., Wang, L., Tang, X.: Detecting doctored JPEG images via DCT coefficient analysis. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3953, pp. 423–435. Springer, Heidelberg (2006). doi:10.1007/11744078_33
9. Adobe Systems Incorporated: Digital negative (DNG) specification, version 1.4.0.0 (2012)
10. Julliand, T., Nozick, V., Talbot, H.: Automated image splicing detection from noise estimation in raw images. In: Imaging for Crime Prevention and Detection, pp. 1–6 (2015)
11. Lin, Z., He, J., Tang, X., Tang, C.: Fast, automatic and fine-grained tampered JPEG images detection via DCT coefficient analysis. Pattern Recogn. **42**(11), 2492–2501 (2009)
12. Lukáš, J., Fridrich, J., Goljan, M.: Detecting digital image forgeries using sensor pattern noise. In: Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII, vol. 6072, pp. 0Y1-0Y11 (2006)
13. Mahdian, B., Saic, S.: Detection of resampling supplemented with noise inconsistencies analysis for image forensics. In: International Conference on Computational Sciences and its Applications, pp. 546–556, July 2008
14. Mahdian, B., Saic, S.: Using noise inconsistencies for blind image forensics. Image Vis. Comput. **27**, 1497–1503 (2009)
15. Pan, X., Zhang, X., Lyu, S.: Exposing image forgery with blind noise estimation. In: The 13th ACM Workshop on Multimedia and Security, Buffalo, NY (2011)
16. Pan, X., Zhang, X., Lyu, S.: Exposing image splicing with inconsistent local noise variances. In: International Conference on Computation Photography (ICCP), pp. 1–10, April 2012
17. Popescu, A.C., Farid, H.: Statistical tools for digital forensics. In: 6th International Workshop on Information Hiding (2004)
18. Popescu, C., Farid, H.: Exposing digital forgeries in color filter array interpolated images. IEEE Trans. Signal Process. **53**(10), 1948–3959 (2005)