Automatic noise-based detection of splicing in digital images

Thibault Julliand

Automatic noise-based detection of splicing in digital images

Thibault Julliand

Université Paris-Est Marne-la-Vallée

A thesis submitted for the degree of

Doctor of Philosophy

19 January 2018

Hugues Talbot	*Directeur de these*
Christine Fernandez-Maloigne	*Rapporteur*
William Puech	*Rapporteur*
Vincent Nozick	*Examinateur*
Florent Retraint	*Examinateur*
Jacques Blanc-Talon	*Examinateur*

Acknowledgements

I am extremely honored to have had the opportunity to work with my thesis advisors, Hugues Talbot and Vincent Nozick, and I would like to thank them for the help and support they have provided me over the last three years. Hugues was the one who got me interested in digital forensics, and his past experience in that domain proved invaluable to kickstart my research. Vincent got me up to speed in computer vision and programming, and was an excellent guide when I started venturing further in the forensics domain. I am sure that their patience was tried repeatedly over the course of this thesis, and I thank them again for sticking with me despite that.

In addition, I would like to thank Diane Genest, Ali Kanj, Julie Robic, Odyssée Merveille, Eloïse Grossiord, Clara Jacquet, Laurent Noël, Matthieu Lagarde, Geoffrey Giry, and all the folks at the LIGM lab for the support and team spirit they shared with throughout. A special thank for Elodie Puybareau for all the help she provided on the various administrative matters inherent to a thesis.

Thanks also to my flatmate, Vincent Bonnemains, for tolerating my stress levels before big deadlines and providing me with sustenance during the writing of this manuscript.

I would also like to thank my close friends outside the lab, Idris Weber, Maël van Beek, Claire Gremy, and Rémi Bufnoir, for their frequent, though most often inadequate inputs on how to solve my research problems. Their friendship and support helped to carry me through the work.

I also offer a thank to my various online friends, through various supports, for the laughs and relaxation they helped me get during the last few years.

Finally, I would like to thank my family, and especially my parents, Eric Julliand and Laurence Ohmann, for their love and support. They have made a lot of sacrifices for me, and have provided me with encouragements every step of the way. I would like to dedicate this dissertation to my grandmother, Bernadette Julliand.

Abstract

In this dissertation, we offer three new forensics imagery methods to detect splicing in digital images by exploiting image noise statistics. To do so, we introduce a new tool, the noise density histogram, and its derivative, the noise density contribution histogram. Our methods allow splicing detection on both raw and JPEG images. Although the use of noise discrepancies to detect splicing has already been done multiple times, most existing methods tend to perform poorly on the current generation of high quality images, with high resolution and low noise. The effectiveness of our approaches are demonstrated over a large set of such images, with randomly-generated splicings. We also present a detailed analysis of the evolution of the noise in a digital camera, and how it affects various existing forensics approaches. In a final part, we use the tool we developed in a counter-forensics approach, in order to hide the trace left by splicing on the image noise.

Contents

List of Figures

A picture is worth a thousand words.

— Arthur Brisbane [1]

1 Introduction

Contents

1.1 Motivation

The first digital camera, the MegaVision Tessera, was commercialized in 1987. Since then, the world of digital photography has been in a state of constant expansion and improvement: longer battery life, increased resolution, automated focusing and white balance, various modes for every kind of situation, etc. Every smartphone now includes a digital camera. As a consequence, the world has moved away from film-based pictures and wholly adopted digital images. However, this rising trend was accompanied by a darker side: image falsification.

Photography falsification is older than the digital age: for example, censorship or alteration of images was a widespread practice in the USSR [2], as shown in Fig. 1.1. Those practices, however, required a trained hand a special equipment. With digital images,all that is needed is a computer and an image-processing software, the latest being commercially available, or even free. As such, almost anyone can alter a digital photograph. In order to combat this new form of misinformation, the field

Figure 1.1: Successive alterations of a portrait of Soviet leadership as members fell out of favor with Stalin.

of digital forensic developed: a way to detect, through various attributes of digital images, all kind of falsifications. The work presented here aims to provide new tools to detect a particular type of falsification called splicing, using information provided by the image noise. Splicing, also called "exogenous insertion", consists in inserting part of an image A into another image B, as shown in Fig. 1.2.

Figure 1.2: Contrarily to what this image could imply, the author is not Batman.

1.2 Contribution

The central contribution of this dissertation is three new tools to detect splicing in digital images. It also includes an in-depth analysis of image noise and counter-forensics method designed to hide falsifications.

Chapter 3 offers a detailed study of noise in digital images, from its sources to the final image. In a first part, the effect of the different steps of the camera internal processing (demosaicking, internal denoising, white balance, contrast, luminosity, compression, etc) is analyzed, with a comparison of the various possible options. Then, we study the impact of different "classical" image processing methods

(resampling, additional compression, rotation, etc) on the image noise. Finally, the last section looks at the effect of strong falsifications on the image, as well as the robustness of various forensic approaches to added noise.

Chapter 4.2 proposes a first approach of splicing detection, applied specifically to raw images, which are the ones we get directly out of the sensor. This method is based on the fact that the raw format contains Poisson-Gaussian noise, contrarily to the noise in a JPEG image which is usually considered to be a white gaussian noise. Additionally, the raw images have the advantage of being free of any processing, and as such they have a "pure" noise. Based on these assumptions, the proposed method divides the image in tiles and estimates the noise in each one in order to identify potential anomalies, by clustering the tiles according to their noise characteristics.

Chapter 4.3 introduces the basis of the core tool developed in this thesis, the noise density contribution histogram, along with a first application. As in the previous chapter, this application will be used on raw images. The proposed tool is based on building the noise density function of the image (specifically, a representation of it as an histogram). If we assume that a spliced element has a different noise compared to the rest of the image, then it will contribute to certain specific areas of the noise density function. By observing which pixel clusters contribute to those specific areas, we can isolate the spliced element from the original image.

Chapter 4.4 extends and polish the method presented in the previous chapter, by applying it to the JPEG format. In order to adapt to the limitations imposed to the accuracy of the noise in JPEG, we transform our original contribution histogram, sparse in nature, in a dense version. The base form of this histogram is then used to classify our tiles in two categories.

Finally, Chapter 5 offers an application of our tool to the domain of counter-forensics, which is the field dedicated to hiding alterations in an image. By knowing the noise characteristics of an image, we can reproduce them on the spliced element in order to make its noise virtually identical to the one in the rest of the image. An alternative application is proposed in making synthetic images more realistic.

Ipse se nihil scire id unum sciat

I know one thing; that I know nothing.

— Cicero's *Academica, Book I, section 16*

2

Digital Image Forensics Overview

Contents

2.1 Distinction Between Forgery And Enhancement

It is rather normal, nowadays, to perform minor aesthetic enhancement on an image to improve its overall visual quality. The development of things like Snapchat or Instagram filters, to apply various kinds of effects on a picture is a good example of that, like in Fig. 2.1. However, those enhancements still alter the image and as such, it is important to know where the distinction is between image improvement and image forgery [3]. First of all, most "radical" modifications, such as splicing or cloning a large area, can be immediately considered as forgery. The line blurs

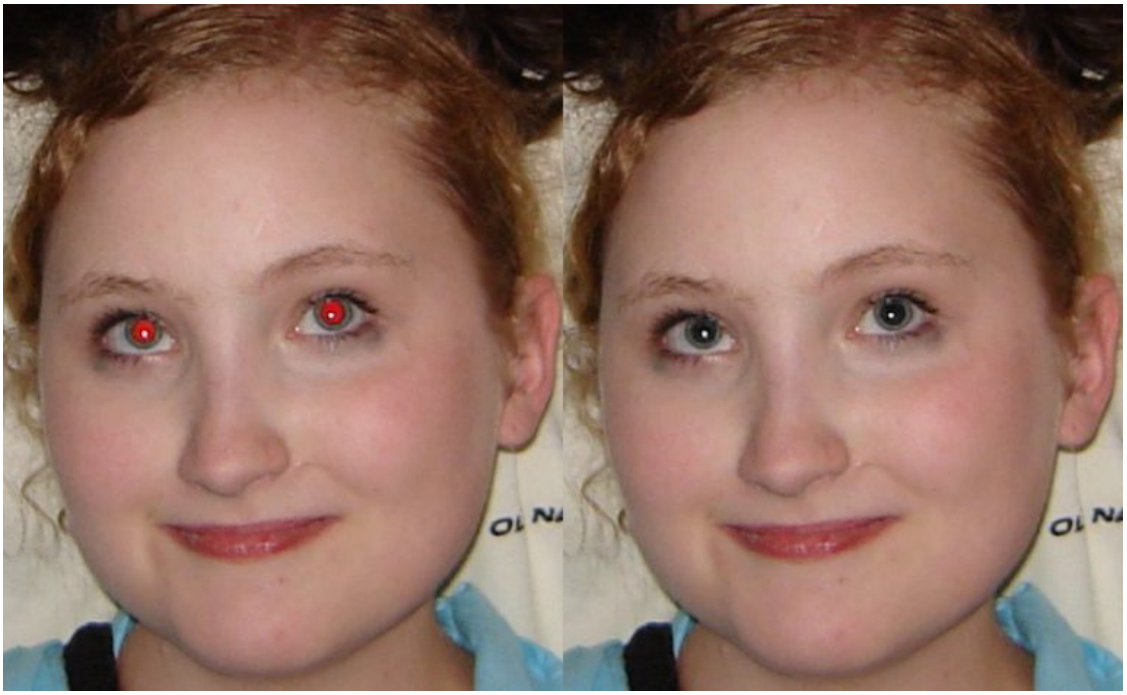


Figure 2.1: A purely aesthetic image modification. Although it is possible that the subject eyes were originally red, it is unlikely.

when it comes to operations like contrast or chroma change. Although they are usually seen as mere improvement tools, they can still be used in order to obfuscate or erase elements of an image. For example, increasing the overall luminosity of an image to bring certain parts to saturation can effectively hide previously visible elements, an example of which can be seen in Fig. 2.2.

In the same way, changing the chroma key in a specific region of an image can be used to make an item or person less distinguishable from the rest of the image. As such, in those cases, the main element differentiating forgery from enhancement is mainly intent; something which, unfortunately, can not be judged simply by analyzing the image.

2.2 Main Forgeries And Counters

With the distinction between forgery and enhancement established, we will examine the two main types of forgeries, colloquially known as copy-rotate-move and splicing, and the methods that have been developed to counter them. We will also look

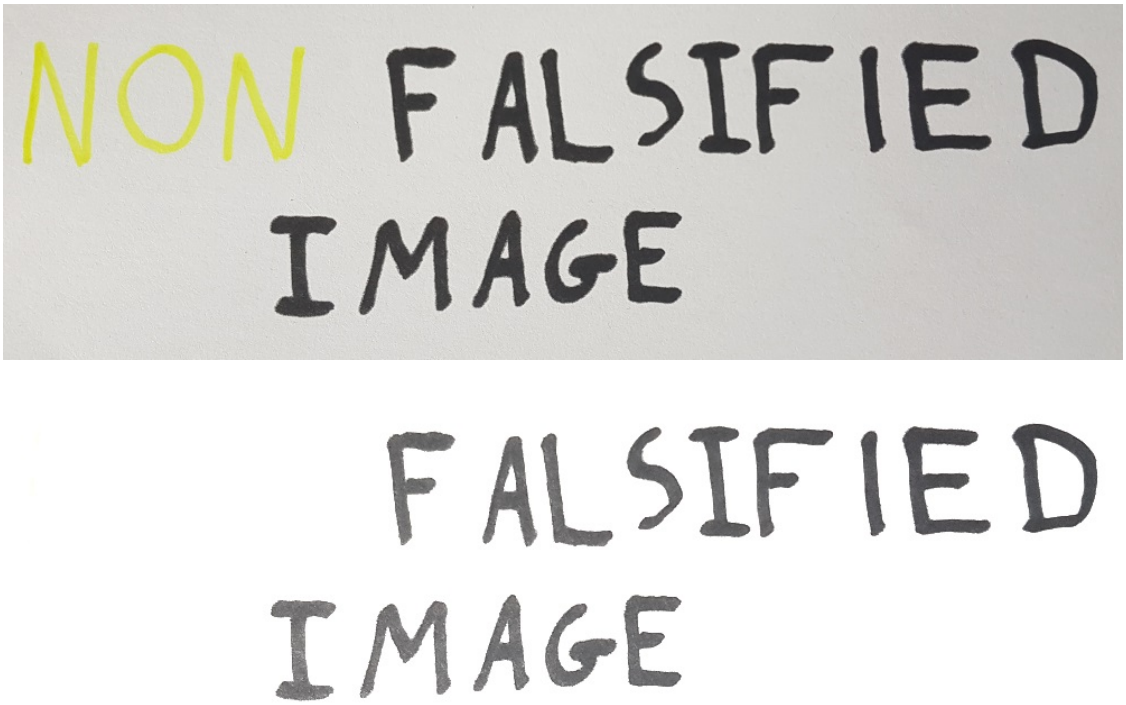


Figure 2.2: An example of "soft" forgery. The only difference between the two images is brightness and contrast, and yet some critical piece of information has disappeared.

at format-based detection methods, which exploits specificities of the files format (usually JPEG) to expose forgeries.

2.2.1 Copy-Rotate-Move Detection

The copy-rotate-move forgery, formally endogenous insertion, consists in cloning - and potentially rotating or resizing - part of an image to insert in the same image. This method is commonly used to increase the size of a set of elements, like a crowd or, more recently, a missile battery test [4], as shown in Fig. 2.3.

Most copy-move forgery detection algorithms (CMFD) follow one of two approaches: they're either keypoint-based or block-based. Keypoint methods have mainly been exploiting variants of the SIFT (Scale-Invariant Feature Transform) algorithm since its proposed use by Huang et al. [5], with improvements proposed over the years to increase robustness against compounded alterations. SIFT identifies an object by associating it with a feature vector, assigning it keypoints and distance

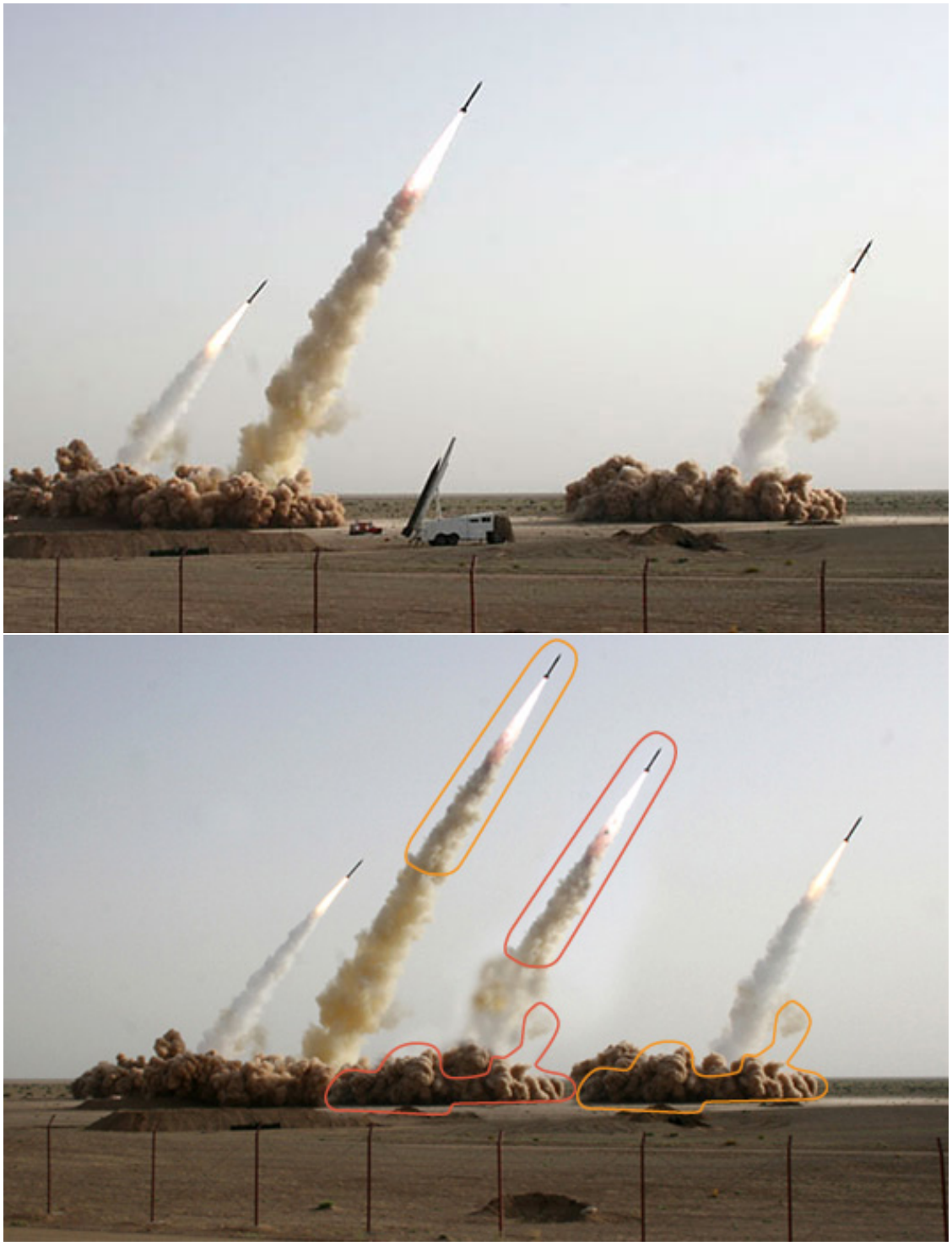


Figure 2.3: An example of copy-rotate-move forgery. The orange zones are original, the red ones are duplicated.

between those keypoints. From there, it is possible to search the image for the same feature vector, or one which has undergone geometric transformations (rotation,

resizing). In other words, SIFT lead to a set of point correspondences from the image to itself, and these points are clustered in the transfomation space (i.e. the points that are linked by the most encountered transformation). For example, Pan and Lyu [6] approach has been optimized to better handle rotations and mirroring, with results shown in Fig. 2.4.

Block-based approaches tend to vary more in the elements they're using: Bayram et al. [7] do a bit-wise block analysis, called a Binary Similarity Measure. This method considers each bit plane of an image as its own binary pictures, and analyzes correlations (or lack thereof) to highlight the forgery. Kang and Wei [8] use a singular value decomposition of the blocks to extract features of interest, as in a keypoint-based approach, and Ryu et al [9] exploit the Zernike moment to locate similarities. The Zernike moment is an invariant function that can be used to extract features from the image, and is robust to various post-forgery alterations, such as added noise. The features of each block are extracted, then sorted lexicographically. Two blocks are considered forged if their Euclidean distance is too small. An example of their results is shown in Fig. 2.5.

2.2.2 Splicing Detection

The splicing forgery, formally exogenous insertion, consists in taking an element of an image and inserting it in a different image. It can be used in much the same ways as a copy-move operation, but also to completely alter the information of an image [10], with an example in Fig. 2.6.

The methods used to detect splicing cover a wide specter, exploiting different elements of the image, but can be divided into a few broad families. The first one is format-dependent, mainly exploiting the particularities of the JPEG format: for example, the JPEG quantization analysis by Popescu et al. [11], He et al. [12], and Lin et al [13] tries to detect different rates of compression throughout the image. Those methods decompose the image into its DCT components and, combined with its quantization table, build histograms of the main frequencies on block version

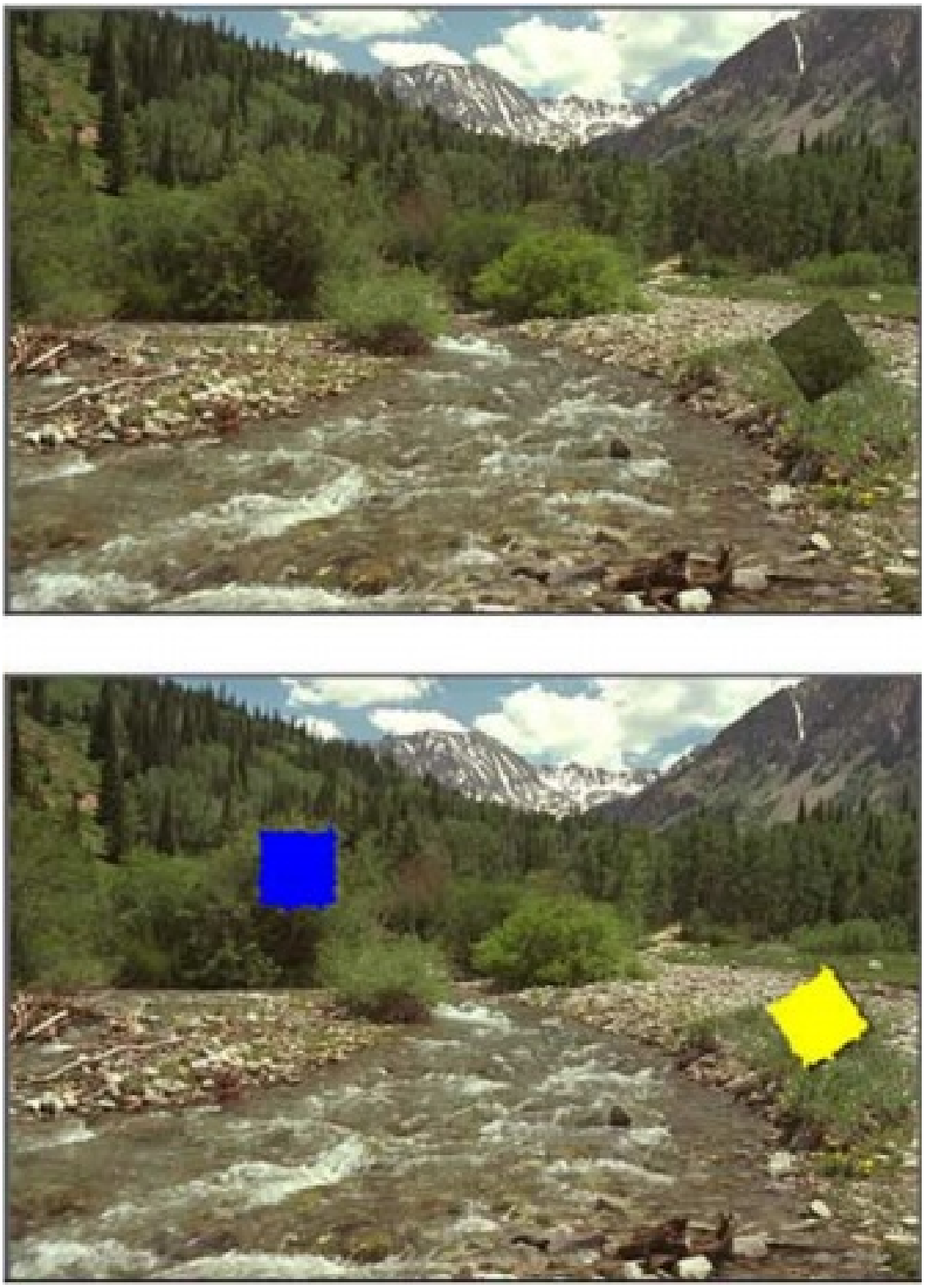


Figure 2.4: Pan and Lyu [6] CRM detection results. The colored squares on the second image indicate the matched areas.

Figure 2.5: Ryu et al [9] CRM detection results. The bottom plane is a duplication, and the suspect areas are highlighted in white in the second image.

Figure 2.6: An example of splicing forgery. The two top images have been merged to produce the bottom one, which has a very different feeling.

of the base image. This gives a probabilistic result, with possibly tampered or genuine regions. Lin et al. results are shown in Fig.2.7.

A second category is based on the use of the Color Filter Array (CFA), whose functioning is detailed in Section 3.3.1.2. A representation of an unaltered CFA

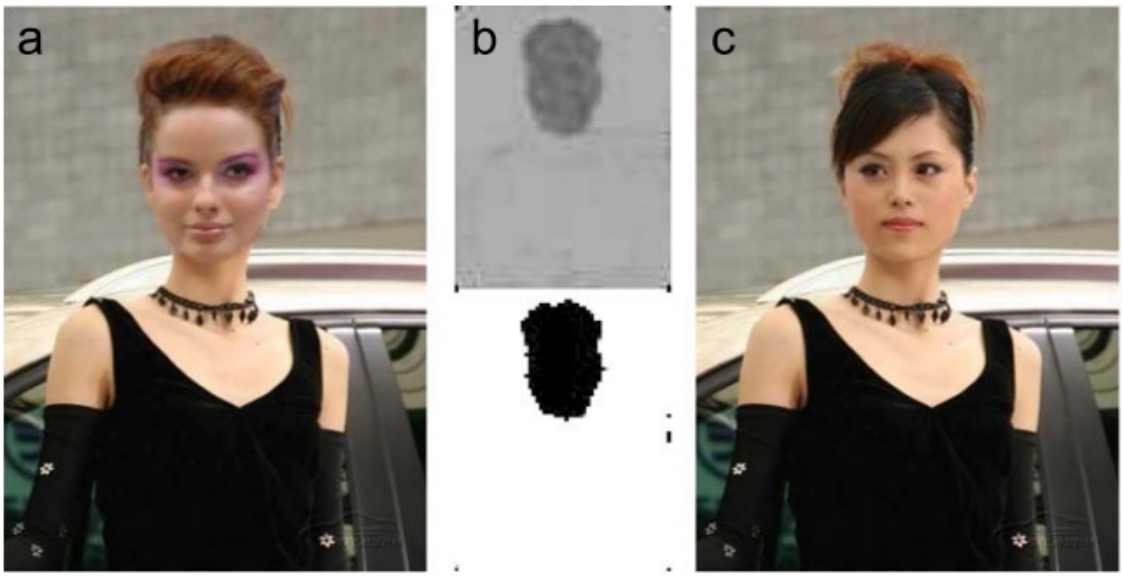


Figure 2.7: Lin et al. [13] results. The original image is on the right. The middle image shows the areas where the algorithm has found a probable alteration.

is presented in Fig. 2.8. Such methods are presented by Popescu and Farid [14], Gallagher and Chen [15], or Ferrara et al. [16]. These three methods use irregularities in the expected CFA interpolation pattern to highlight suspicious areas in the image. Ferrara et al. results are shown in Fig. 2.9.

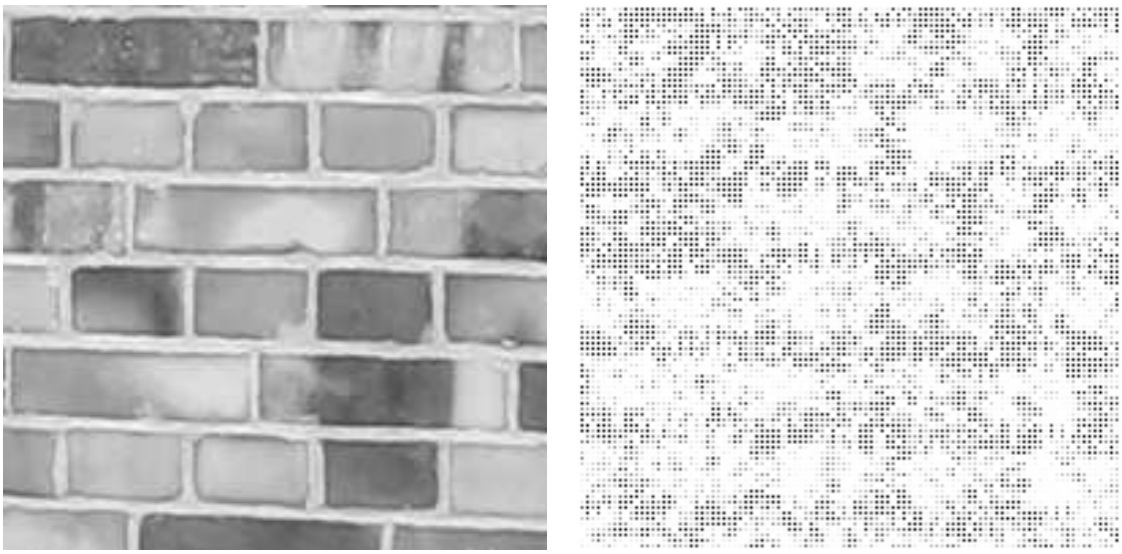


Figure 2.8: An example of CFA from an unaltered image. The grid pattern on the right image shows no obvious disruption. Image from [17].

A third approach is based on machine learning and statistical analysis, such as presented by Bayram et al. [18], Fu et al. [19], or Han et al.[20]. Those methods use

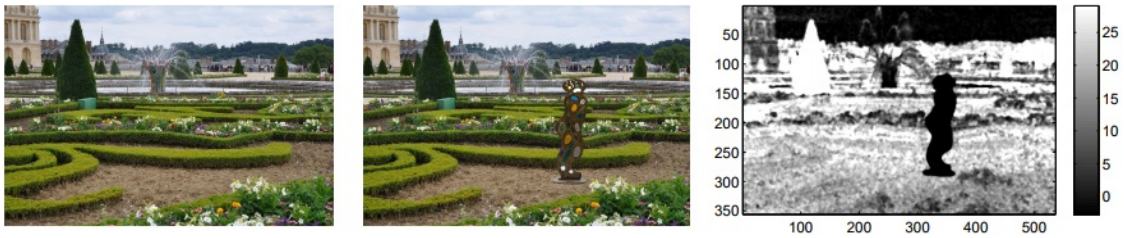


Figure 2.9: Ferrara et al. [16] results. The original image is on the left. The right image shows the areas where the algorithm has found a probable alteration. We can notice that the sky is considered as suspect as the spliced element. This is due to the difficulty to get a correct CFA interpolation pattern in bright, untextured areas.

support vector machines (SVM), coupled with large image databases, to extract of set of statistical values and characteristics that are specific to altered or genuine images. Although those methods tend to be extremely efficient, they require an important number of images to extract the desired characteristics. The fourth category exploits geometrical and lighting informations, as shown by Johnson and Farid [21, 22] or Chennamma and Rangarajan [23]. Those approaches tend to slightly underperform compared to other categories, due to the importance of context and semantic clues in the analysis of geometry and lighting. Johnson and Farid results are shown in Fig.2.10.

2.2.3 Format-based detection

Some methods can be to detect both types of forgeries: the double JPEG detection method introduced by Popescu and Farid [11] is based on an effect of the quantization step of the JPEG compression. The process of compressing twice in JPEG changes the quantization histogram in an easily recognizable way as shown in Fig. 2.11. Although this method is very hard to cheat, it does not indicate a falsification per se, merely that the image has been saved twice as a JPEG.

The JPEG Ghost method presented by Farid [24] is an extension of the double JPEG that handle local properties of the image. It searches for differences in JPEG quality throughout the image, "quality" being the level of quantization (and as such compression). An example of their results is shown in Fig. 2.12.

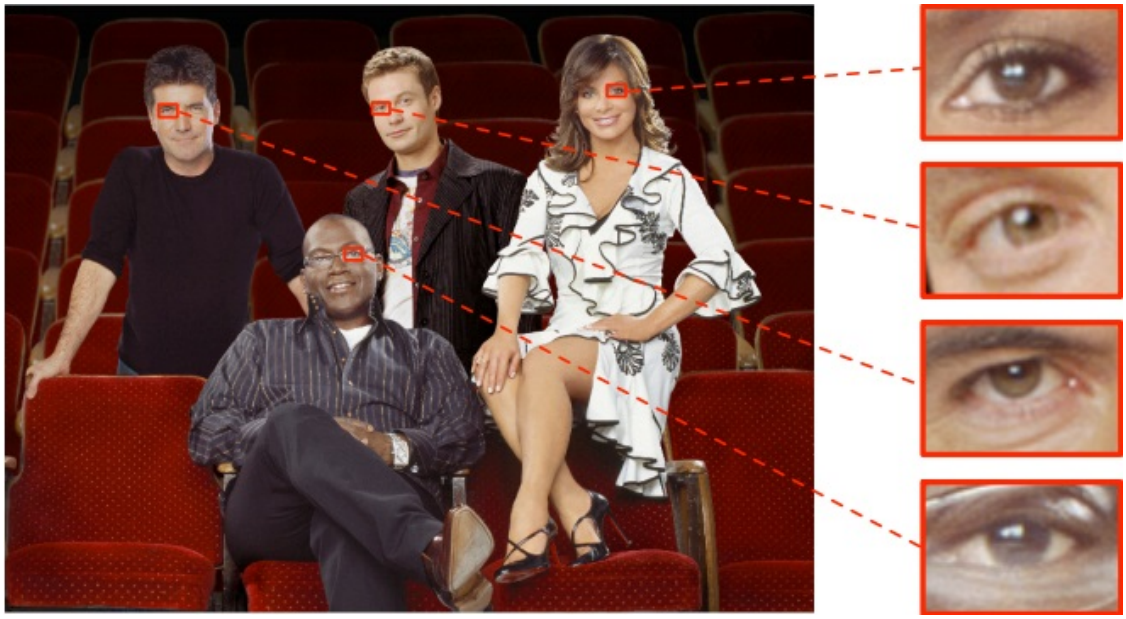


Figure 2.10: Johnson and Farid [21] results. The images on the right show that the eyes of the people in the image do not all show the same light reflection (notably the third one), which indicates a falsification.

Some histogram-based methods are used to detect global pixel intensity modifications, typically from Look-up tables (LUT). Stamm et al. [25] detects the residual peaks of the image histogram resulting from such transformation, by analysing the frequency spectrum of the histogram. Finally, some methods detect very specific alterations: Farid [24] offers an approach to detect resampling (resizing) in an image, by looking for the periodic correlation between pixels introduced by the resampling process using an EM approach. The result image shows a very distinct grid-like pattern, as shown in Fig. 2.13

2.3 Splicing Detection State of the art

The methods presented up until now have been general approaches to falsification detection. In this section, we will concentrate on methods which exploit the image noise in order to detect splicing. As discussed in Chapter 3, the noise in a digital image carries a lot of information. As such, numerous methods have been developed to detect splicing by exploiting different aspects of the image noise. Indeed, it can be safely assumed that the overall noise will be consistent inside of a single

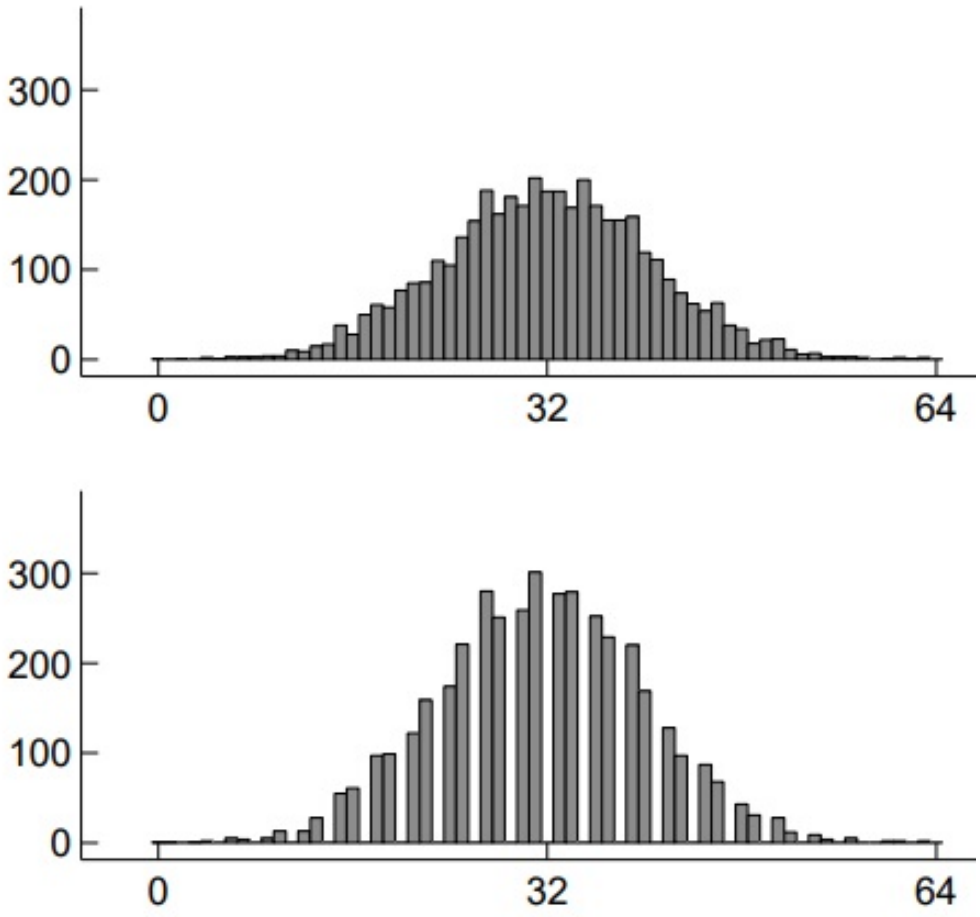


Figure 2.11: Popescu and Farid [11] results. The top image is a single-saved JPEG, the bottom is double-saved. The difference is very easy to detect automatically.

image, and is dependent on the camera model, the lighting conditions, shutter speed, etc. It can be induced that the noise in two different images will tend to differ somewhat, and as such a variation of noise parameters in a specific zone of a single image is a strong indicator of alteration or splicing.

2.3.1 PRNU and camera fingerprint

The PRNU (Photon Response Non Uniformity) can be considered as the "fingerprint" of a digital camera. As explained in Sec. 3.2.1, the PRNU is caused by individual defects in the wafers of the camera sensor. As such, they produce a specific noise pattern on every image taken by a single camera. An example is shown in Fig. 2.14 (averaging many denoised images from the same camera can be enough to estimate

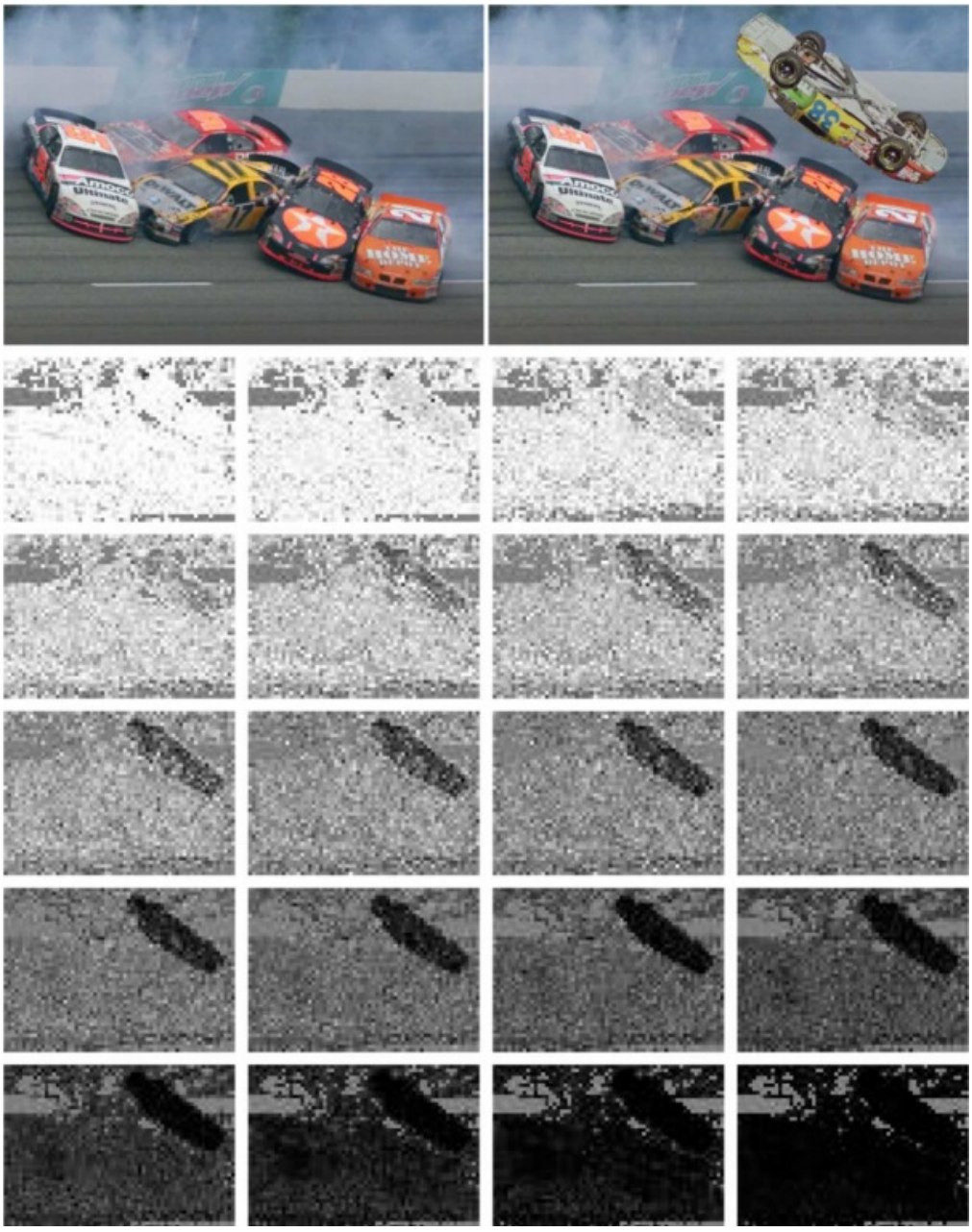


Figure 2.12: Farid [24] results. The result images show the level of quantization in the image though different levels of compression, highlighting the spliced element.

the PRNU). Several methods have been developped to exploit the PRNU and the camera fingerprint: Lukáš et al. [54] propose to check an image against the reference pattern of the camera that took it. Although this method is extremely efficient, with results shown in Fig. 2.15, it requires to have the camera that took the picture on hand, or several pictures taken by this camera. Although there are some contexts where this prerequisite is a relatively same assumption (a court of

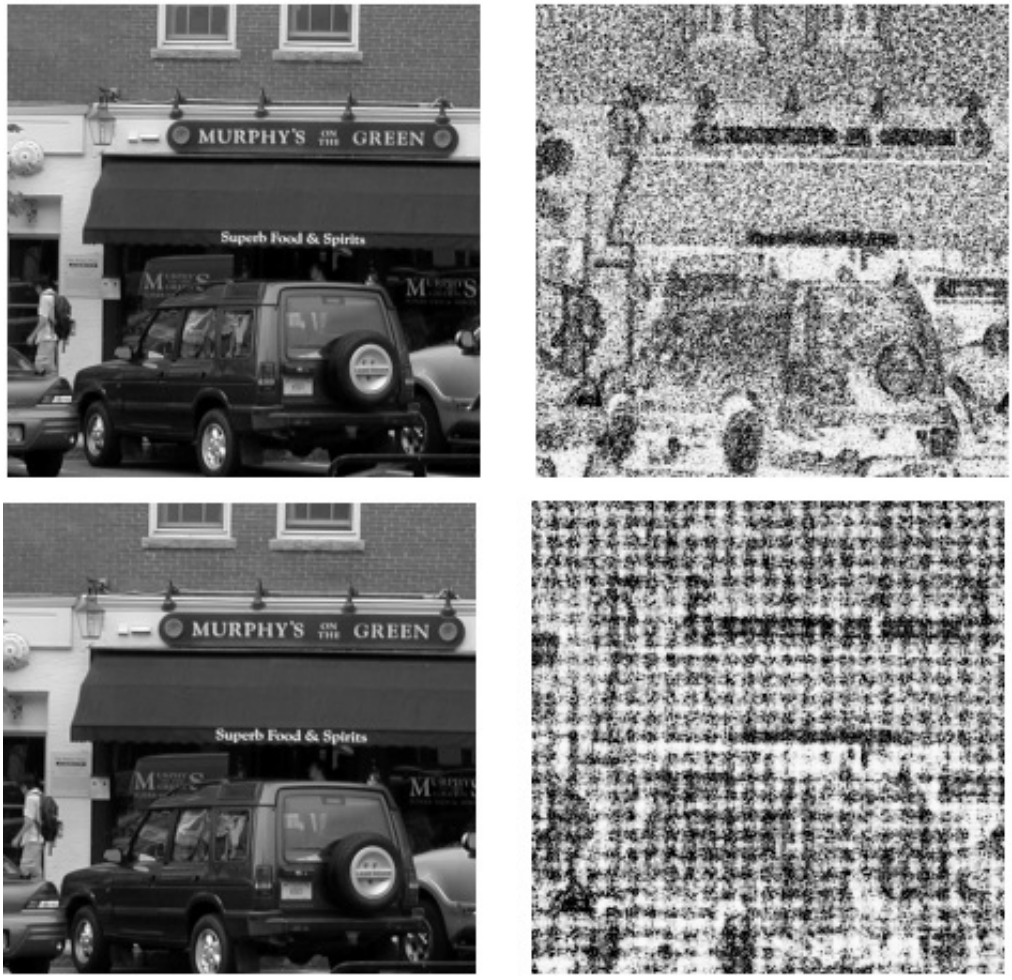
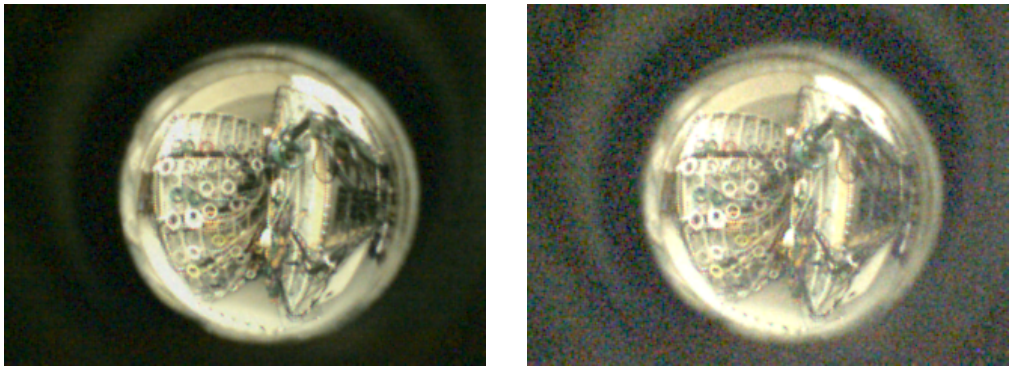


Figure 2.13: Farid [24] results. The second image size has been augmented by 5%, and we can see the grid pattern on its result image.

law, for example), it cannot be truly considered as a blind approach. Chierchia et al. [55] propose a more advanced method which can extract the PRNU from a single image, and verify its statistical consistency with a block-based approach. This allows to highlight inconsistent areas and probable forgeries. Although this method seems fairly reliable, the PRNU approaches overall are subject to several issues: first, the PRNU is a rather weak noise compared to the other noises in an image (most notably the Gaussian noise). Second, the PRNU noise is increasingly getting removed by the camera itself using internal software [56] or in factory.



(a) Base Image. (b) Image with the PRNU noise amplified.

Figure 2.14: Example of PRNU in a digital image. The second image PRNU noise has been amplified after being extracted from several images from the same camera. Image from H.G. Dietz [57]

Figure 2.15: Lukáš et al. [54] results. The pedestrian in the first image is a splicing, and is detected in the second image.

2.3.2 Wavelet-based methods

Wavelets are a very powerful too when it comes to noise analysis. Indeed, wavelet decomposition is frequently used in denoising methods since a noise image is usually rather easy to obtain from one of the decomposition sub-bands. Consequently, methods have developed to exploit this tool in digital forensics. Chen et al. [58] use the moment and phase information contained in the wavelets to create a predictive model of what each pixel value should be according to the values of its neighbours. They then classify their results using a trained SVM, with final detection rates higher than 75%. In comparison, Mahdian and Saic [41] perform a block based decomposition of the HH_1 sub-band and estimate the standard noise deviation

in each block. The block are then merged according to their estimated noise, by neighbourhood. This creates a partitioning of the image based on estimated noise value, with the spliced element separated from the rest of the image. Although their experiments only use natural images with noise added in specific areas, and not true splicings, the application to noise-based splicing detection is clear. An example of their results is shown in Fig. 2.16.

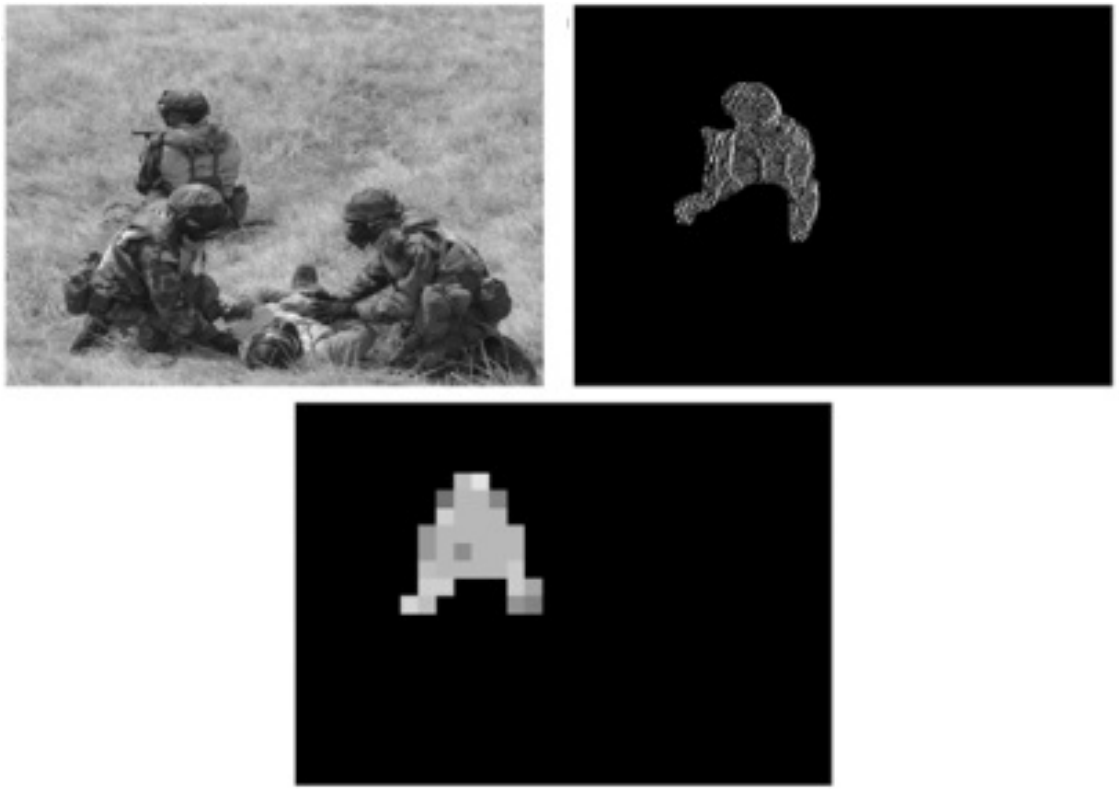


Figure 2.16: Mahdian and Saic [41] results. The top right image shows the noise-corrupted area, and the bottom image is the detected area.

2.3.3 DCT-based methods

The Discrete Cosine Transform (DCT) decomposition is also useful to access an image noise characteristics. The high-frequency coefficients of the DCT of an image are more often than not principal components in the image noise, and their analysis can yield significant results. Pan et al. [42] use a statistical measure called the kurtosis, obtained with the variance and fourth order moment of a random variable, here the noise. They estimate that in a given frequency band, the kurtosis value

of a natural image tend to concentrate around a single value. They compute this value over the whole image, and then repeat the computation at the local scale. This allows them to look for areas where the noise seems significantly different compared to the rest of the image, with results shown in Fig. 2.17. However, their results show that their rate of success is significant only when the noise variance difference between the original image and the spliced element is over 10db. He et al. [59] compute the noise variance based on the DCT coefficient, but they consider the problem as a L1 loss optimization problem, which is linear and therefore easier and faster to solve. This accelerated method allows them to obtain almost pixel-wise result images of the estimated kurtosis and noise variance. Those result images are then clustered using a k-means algorithm to get the final result image, which is shown in Fig. 2.18. He et al. [60] use an approach that exploits both the DCT and the DWT decompositions. However, they use those decompositions to extract Markov features that are analyzed by a SVM. This method has the advantage of using a massive amount of features to distinguish between forged and natural images, thanks to the double source of DCT and DWT transforms, and the multiresolution nature of the wavelets. In order to keep the computational time reasonable, they use a slightly altered SVM called SVM-RFE (support vector machine recursive feature elimination) to reduce the final dimensionality of the feature vector used by the machine. Their approach gives them an accuracy of around 93% on the Columbia dataset [61].

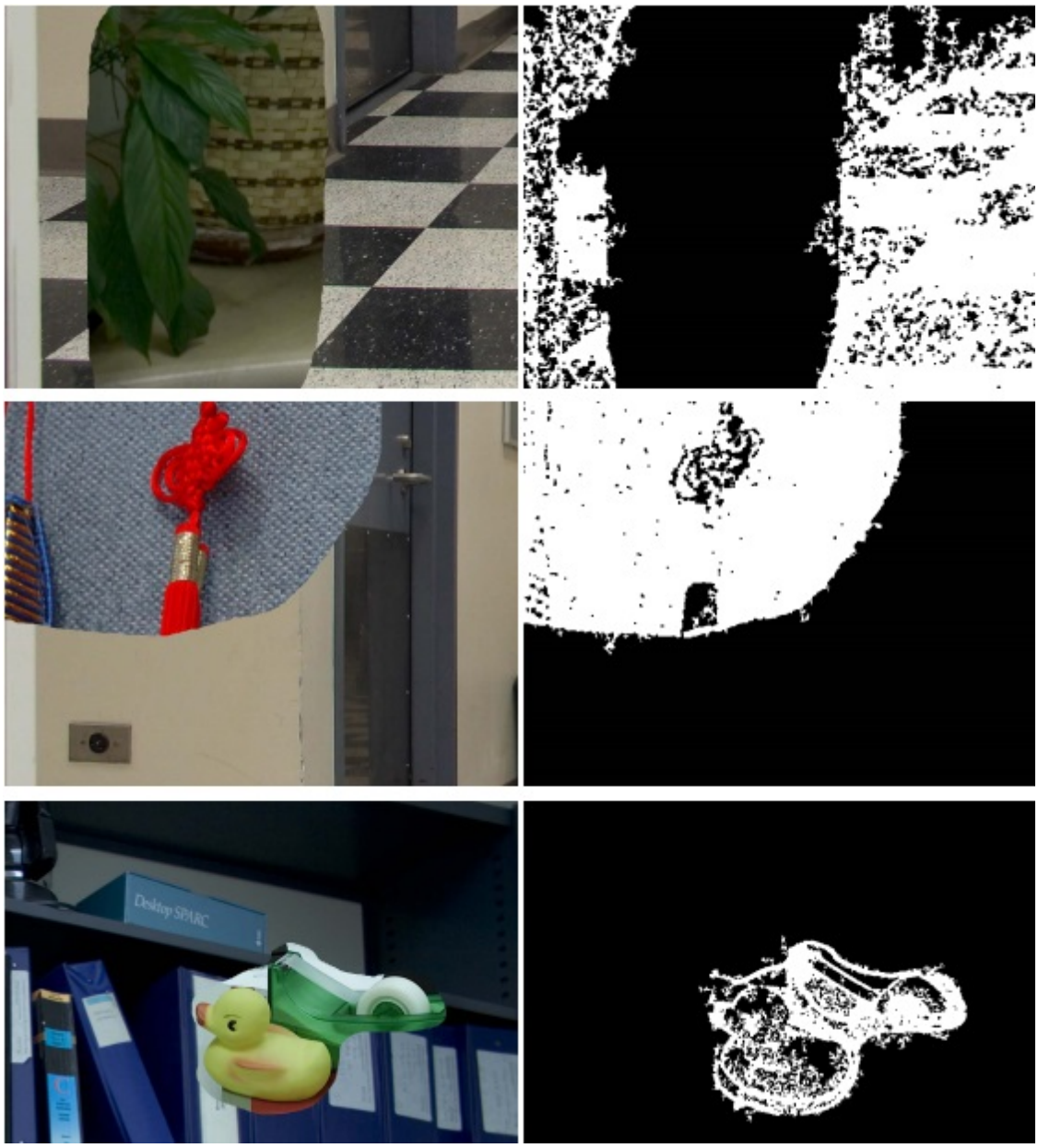


Figure 2.17: Pan et al. [42] results. The right column corresponds to the computed variance. We can note that the top result shows the spliced element in black, because it has a lower noise compared to the original image.

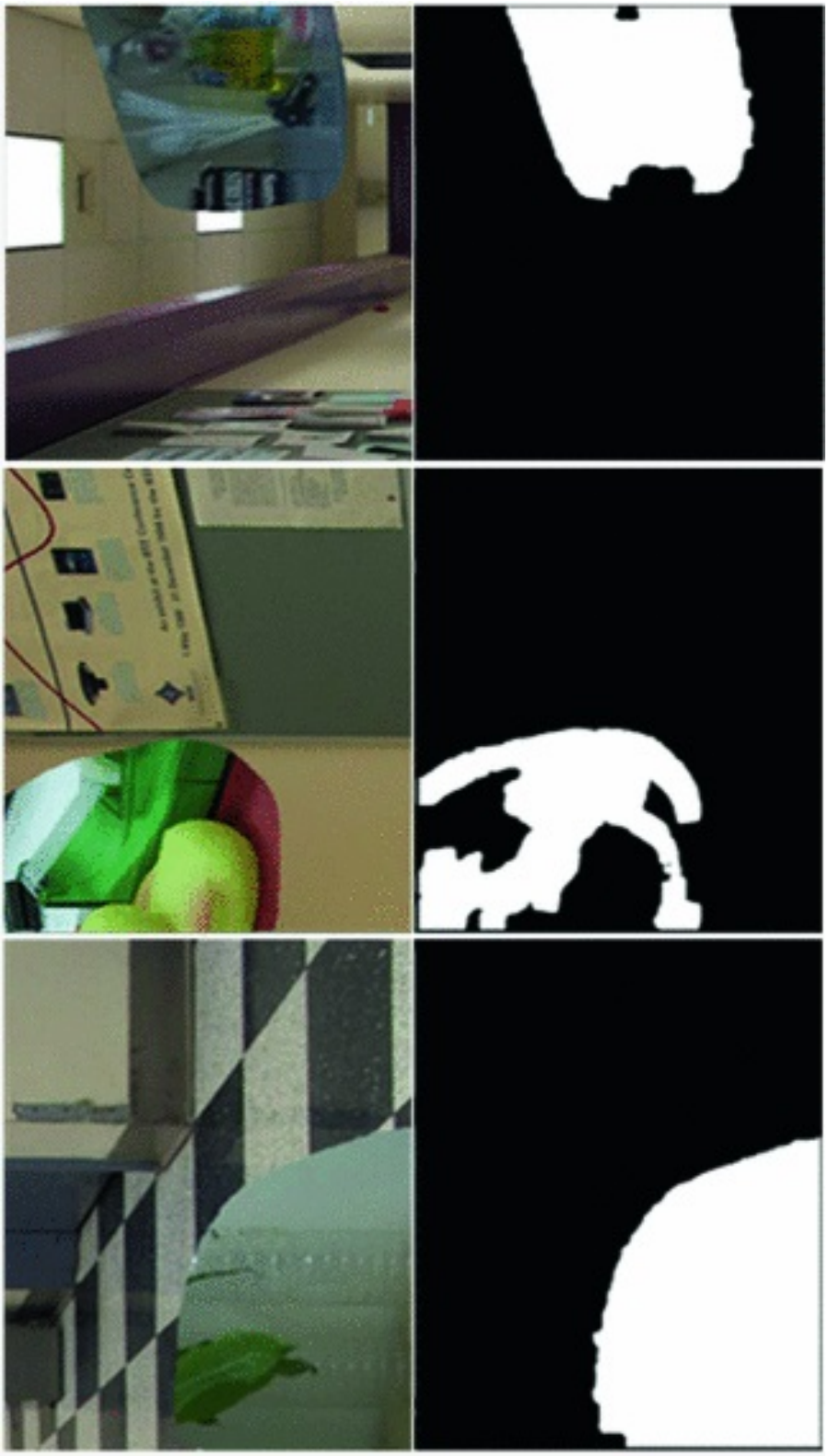


Figure 2.18: He et al. [59] results. The right column corresponds to the final result image obtained after the clustering.

One person's data is another person's noise.

— K.C. Cole

3 Noise Study

Contents

3.1 Introduction

From the film grain of analogue cameras to the sensor noise of digital cameras, image noise has always been a concern in the history of image acquisition. Indeed,

noise is an unwanted artefact that appears during the image capture process and can take various forms depending on the camera model and the lighting conditions during the image acquisition. For aesthetic reasons or for computer analysis clues, many studies have been concerned with suppressing this noise from the "original" signal. This interest is still an active research topic, particularly considering the increasing number of smartphones equipped with low quality sensors.

Since the noise problem is far from solved, some digital image forensics methods attempt to take advantage of this alteration of the signal to detect image forgeries. Some methods focus on the detection of local noise inconsistencies when some others attempt to identify the noise fingerprint of digital cameras.

The purpose of this chapter is to outline the alterations of noise characteristics through the camera pipeline and the usual post-processing, in order to understand what kind of noise one can expect for image forensics purposes. The content of this chapter was first published at the International Workshop on Digital-forensics and Watermarking 2015 and won the best paper award. The first focus is on digital image noise characterisation and estimation. The next part deals with noise alteration during the image acquisition and processing pipeline. We distinguish the processing inherent to the camera pipeline from the usual post processing available on many software packages. Finally, we present some image forensics methods that fail when the image is corrupted with artificial noise.

3.2 Digital Image Noise

Noise in digital images can come from various sources. Some are physical, linked to the nature of light and to optical artifacts, and some others are created during the conversion from electrical signal to digital data. As noise degrades the quality of an image, various models have been investigated to modelize the image noise for subsequent reduction or removal, at various steps of the image acquisition process.

3.2.1 Noise Sources

The main sources of noise can be divided into two main categories: the physical noise, linked to physics constraints like the corpuscular nature of light, and the hardware noise, linked to mechanical issues in the camera. Physical noise notably includes dark shot noise and photon shot noise[26]. The dark shot noise is created by electronic fluctuations caused by an accumulation of heat-generated electrons in the sensor. It is related to thermal noise, and can be reduced by cooling down the sensor. The photon shot noise, also called Poisson noise, is the one caused by the corpuscular nature of light: as photons arrive irregularly on the photosites, two adjacent pixels supposed to have a similar value can end up with different photon counts. As the name indicates, the photons follow a Poisson distribution. Its effect decreases proportionally to exposure time.

The hardware noise includes Fixed Pattern Noise (FPN), Photon Response Non Uniformity (PRNU) and quantification noise. PRNU and FPN are caused by imperfections in sensors. For the PRNU, the cause is mainly the inhomogeneity of silicon wafers and light variations in which individual sensor pixels convert light to electrical signals. It is most visible in pictures with a long exposure time and does not follow any particular statistical law. As for the FPN, it is caused by dark currents. Like photon shot noise, FPN tends to be reduced in long exposure images. Both of those effects increase with light intensity. While the FPN can be removed by substracting the dark frame, the PRNU is non-linear and as such is very hard to remove. Quantification noise is caused by the analogic-numeric converter. It is hard to quantify, because the process is non-linear, though there are some accepted models. More advanced analysis can be found in [27, 28].

3.2.2 Noise Model

In the literature, the overall noise produced by a digital sensor is usually considered as a stationary white additive Gaussian noise. In [29], Faraji et al. justify the use of the Gaussian model for a specific interval of light intensity. However, this approach tends to overlook several noise components, even if we consider it in a

global perspective. Jezierska et al.[30] present a more robust model which also considers a Poisson component. Both of those models take a high-level approach, trying to offer a simplified overall model. In [31], Irie et al. present another approach, which consists in modelizing the noise step by step to get to a final formula. While this approach gives extremely precise results, it requires some specific data, such as the gain parameter for digital image enhancement, and thus cannot be used for blind modeling.

In the following parts of this section, we adopt the Poisson-Gauss model from [30]. This model is applied to all the pixel s of the image with :

$$I_s = \alpha Q_s + N_s \tag{3.1}$$

where Q_s is analogous to a Poisson distribution $\mathcal{P}(u_s)$ of the "clean" signal u_s and $\alpha \in \mathbb{R}$ is a scaling parameter corresponding to the strength of the Poisson component in the noise. N_s is analogous to a Normal distribution $\mathcal{N}(c, \sigma^2)$ with mean $c \in \mathbb{R}$ and standard deviation $\sigma > 0$.

3.2.3 Noise Estimation and Denoising

Image denoising is a very active research field in the signal and image processing community. However, most existing denoising methods require noise parameters estimation before denoising. Hence, some noise parameters estimation studies have also been proposed. For accuracy purposes, the following overview presents the methods that can handle both Gaussian and Poisson noise.

Foi et al. [32] present such a method that identifies the Gaussian and the Poisson noise. However, this method is subject to an homogeneous image region search that discards a large part of the pixels of a natural image. Thus it may sometimes fail on small regions of the image where homogeneous parts are too small to be considered. Jezierska et al. [30] distinguish pixels that are more subject to Gaussian or Poisson noise from an iterative Expectation-Maximization process. Nevertheless, this method is extremely slow, and thus is impracticable for regular images. Colom and Buades [33] present a PCA noise decomposition approach which gets fast results and is efficient on post-CFA images.

It is important to note that the main purpose of noise estimators is to get a good noise estimation in order to denoise, but not really to get an accurate noise estimation. Thus, a possible approach for estimating the image noise may consist of first using one of the previously mentioned methods to roughly estimate the image noise, then to denoise the image and then subtracting the result from the original noised image. Among the large variety of denoising methods, the Non-Local Means, proposed by Buades et al. [34], performs well. However our tests show that the method lacks accuracy in highly textured zones. Moreover, this method only denoises the Gaussian noise component. Jezierska et al. [35] follows their previous work [30] and still suffers from time computation issue. Dabov et al. [36] introduce the so-called BM3D algorithm that performs a 3D collaborative filtering. This technique performs high-quality denoising for both homogeneous and textured regions, and denoises both the Gaussian and Poisson components.

For our work, the noise estimation follows the latter approach and the noise estimation is computed from the provided noise image. The Poisson and Gaussian noise parameters are estimated from the difference image between the denoised image by BM3D and the original noised image. We first divide the pixel luminance range into n equal intervals $\mathcal{I}_i$, $i \in [1, n]$. The pixels of the denoised image with intensity in $\mathcal{I}_i$ are grouped together to compute a variance σ_i of these pixels in the noise image and a mean value m_i in the denoised image. As specified by [32], the noise that appear in the lower and higher pixel intensities is not reliable. Thus, these pixels are discarded from the noise estimation process. The plot of the variance as a function of the mean gives a line which slope corresponds to the Poisson noise parameters and the y-intercept corresponds to the Gaussian noise component. An example of this noise estimation is depicted in Figure 3.1. The intervals $\mathcal{I}_i$ are referred to as *pixel groups* in the following Figures of the section. In the case of a pure Gaussian noise, this line would be horizontal.

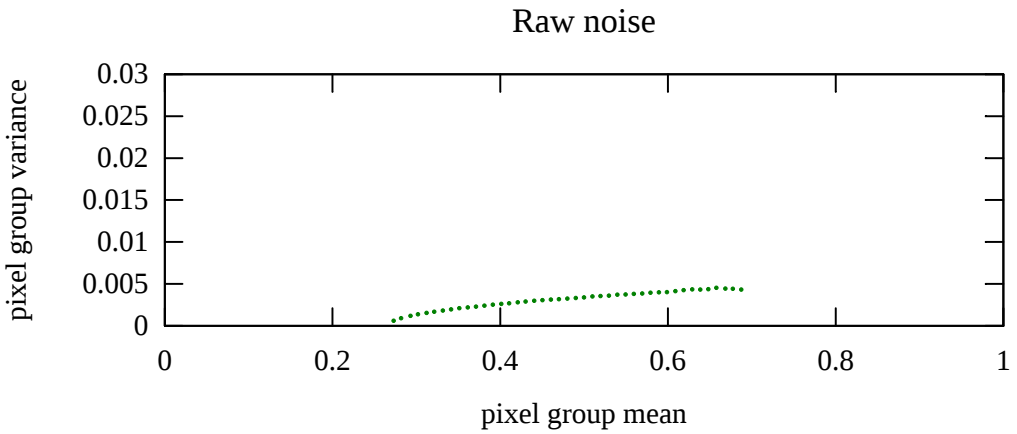


Figure 3.1: Pixel group variances according to the group mean. Here, the (first) green channel of the raw image. This image does not use the full intensity range.

3.3 Noise Alterations Caused by Image Processing

3.3.1 Noise and Camera Pipeline

The camera pipeline often differs from one camera to the next, according to the sensor quality and the camera brand. Differences occur according to the processing steps and their ordering, however most pipelines include similar steps. A good description of these steps can be found in [37, chapters 1 and 3]. In this section, we focus on the processings that affect the noise, and show their impact on a raw image. The tests have been implemented using LibRaw [38] on a set of 15 raw images from various cameras. For clarity purposes, we selected one image (Figure 3.2) of the set with representative results for the figures.

3.3.1.1 Noise Reduction

It is generally preferable to reduce noise as early as possible in the chain, before signal amplifier operations (notably color correction, gamma correction, edge enhancement and color filter array demosaicking). Some standard denoising methods employ wavelet denoising or Fake Before demosaicking De-noising (FBDD) on each of the four channels (R, G1, B, G2). Figure 3.3 shows how a light and a full FBDD affect the image noise.

Figure 3.2: Input image used for the curves computation. *Courtesy of Michel Couprie.*

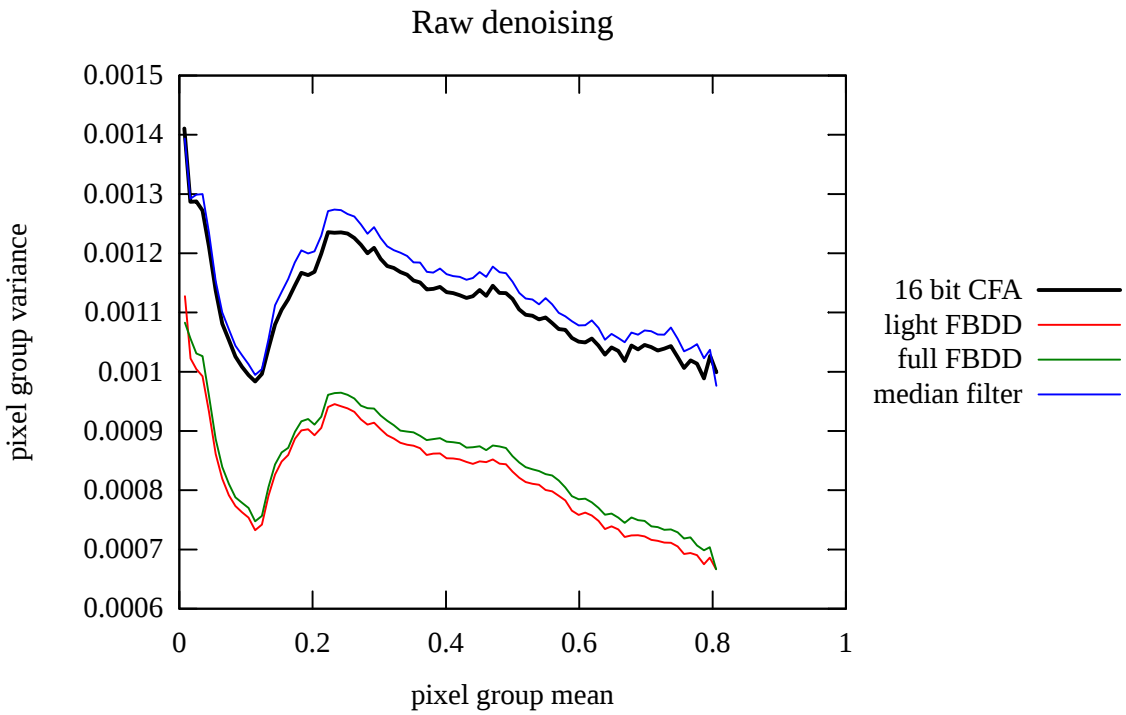


Figure 3.3: Comparison of various denoising methods on raw images. For each image, color filter array demosaicking was performed after the denoising step.

3.3.1.2 Color Filter Array

The Color Filter Array (CFA) allows a single color to be acquired at each pixel. This means that the camera must interpolate the missing two color values at each pixel. This estimation process is known as demosaicking, and modifies the noise properties that could be found on a raw image. Many demosaicking methods also include edge detection or denoising, like Paliy et al. [39]. Therefore, the CFA can have a strong effect on the noise structure, as shown in Figure 3.4, using adaptive homogeneity-directed (AHD) interpolation algorithms of *dcraw*.

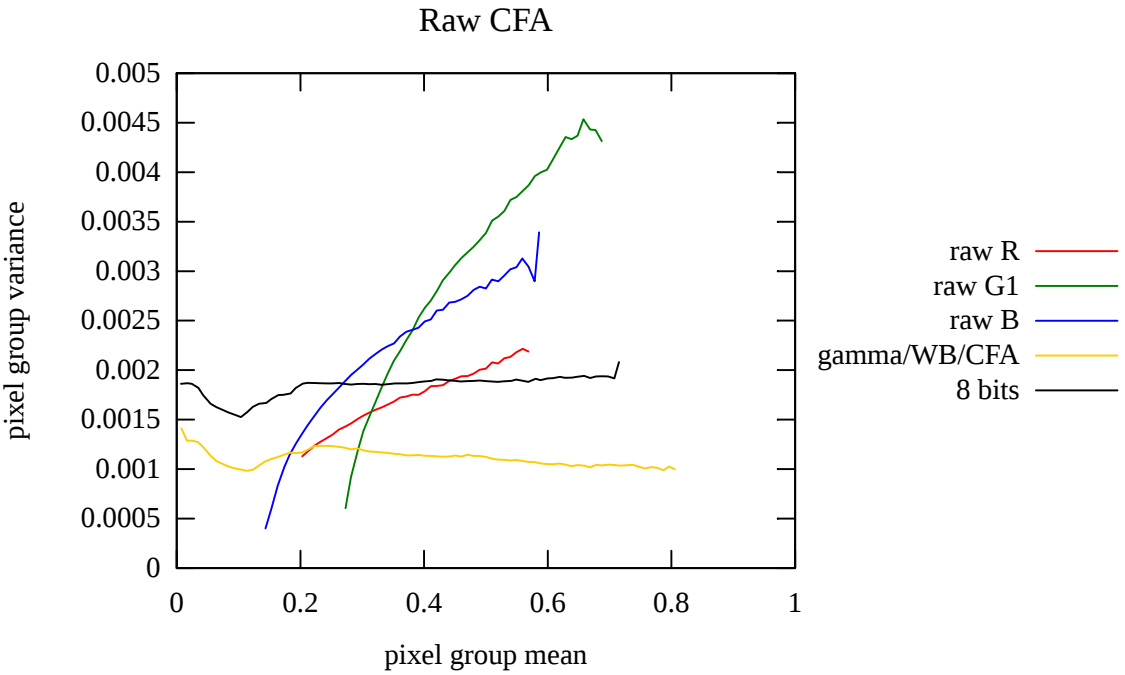


Figure 3.4: This Figure depicts the effect of gamma correction, White Balance and CFA demosaicking on the raw channels of the raw image. The combinaison of these 3 steps has a strong effect on the image noise.

3.3.1.3 White Balance and contrast

White Balance (WB) as well as a contrast operation is just a scaling of all the values of a channel. Since the scaling of a Poisson Gaussian noise remains a Poisson Gaussian noise, the only effect of a WB or a contrast is to enhance or decrease the noise level, however the noise remains present with similar variations.

3.3.1.4 Bit Depth

The conversion for raw depth, usually from 10 bits to 14 bits, to the 8 bits of the usual image file format is a compression that can have varying effects. Intuitively, we could expect noise levels to be reduced. However, we typically observed that the noise was either at the same level, or even higher, after quantification. There are several possible explanations for this phenomenon: first, if the standard deviation is close enough to the conversion quantification step, results can be unpredictable. Second, the quantization actually removes most of the low standard deviation noise, which represents most of the noised pixels, and only leaves the noised pixels with high variations. As a consequence, the calculated standard deviation is much higher, even though the image may look less noisy. This phenomenon can be observed in Fig 3.4.

3.3.1.5 JPEG Compression

JPEG is a lossy compression method with the lossy part predominantly in the high frequencies. Hence, it is not surprising that JPEG compression strongly affect the noise, as depicted in Figure 3.5. However, our tests show that the global shape of the noise is conserved.

3.3.2 "Legal" Image Enhancements

This section presents some usual image filters commonly used to enhance the visual image quality. These processes usually do not involve image forgery. The tests were performed on 8 bits digital images extracted from raw images without a denoising process or lossy compression, i.e. they still contain noise.

3.3.2.1 Image Interpolation

Image interpolation usually results from image resize, which is one of the most common image processing operation. Image interpolation can also occur for many other reasons, such as image rotation, image perspective transformations (e.g. stereoscopic rectification [40]), radial distortion and chromatic aberration correction, etc. The tests have been conducted with bilinear, bicubic, Lanczo and "area" interpolations for both decimation and zoom. The effect on noise is variable

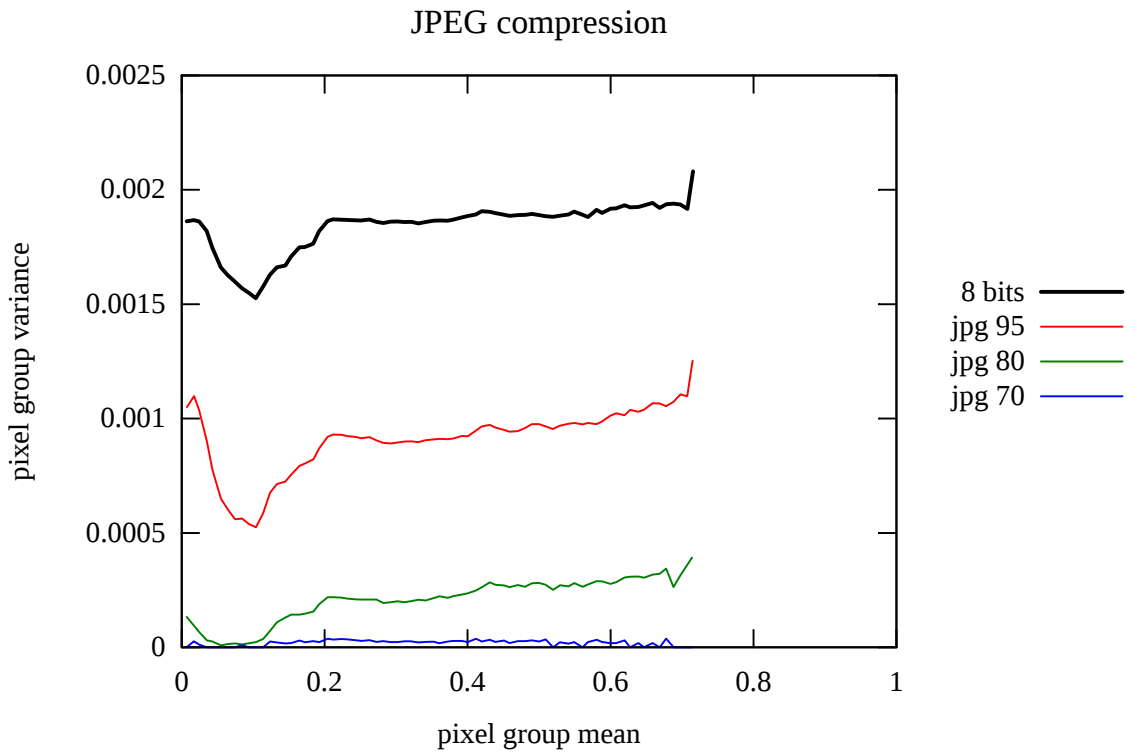


Figure 3.5: JPEG compression effect on noise. The bold line is the uncompressed 8 bits image noise and the other curves correspond to the noise after a compression of 95, 80, 70 and 50.

according to the interpolation method. The noise is always decreased but the global noise shape is globally conserved, as shown on Fig. 3.6.

3.3.2.2 Others

We tested some others images transformations with potential effects on images noise, without significant results. The image saturation process, where colors channels are be mixed together, does not significantly alter the noise. Brightness transformation will just translate the noise to higher pixel intensity levels. Contrast will increase or decrease the noise, but the noise shape remains the same. Image crop will just limit the image surface used for the noise estimation.

3.3.3 Noise and Strong Image Forgeries

This section aims at comparing image noise characteristics after a standard "legal" image enhancement and after a stronger image forgery. The questions are how far the transformed images are from the original in term of noise, and if the noise of

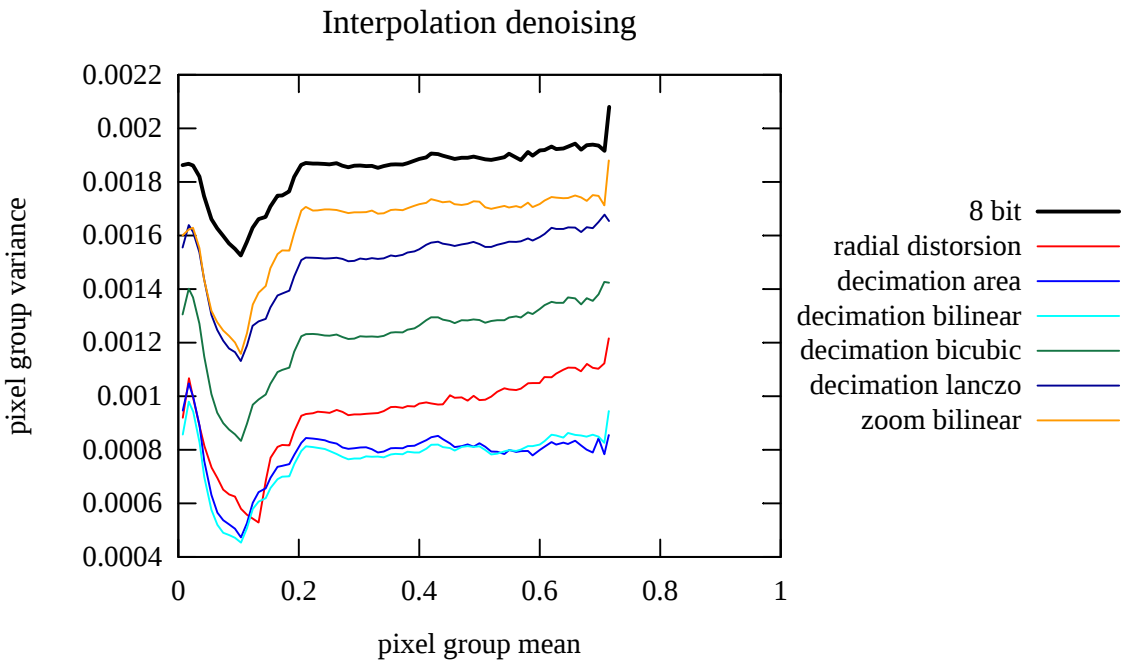


Figure 3.6: Interpolation effect on the image noise.

a strong forgery still makes sens to study. In addition, we denoise the strongly falsified image and renoised it with a light artificial noise.

Figure 3.7 illustrates one of these experiments with the original image, this image with standard image processing (like non-linear histogram manipulations) and the initial image with strong forgeries. Figure 3.8 shows that both soft and strong image forgeries significantly impact the noise. More important, this figure demonstrates how an image with artificial noise may exhibit statistics similar to the initial image.

3.4 Noise and Forensics Detection

This section outlines some digital forensic methods based on noise analysis.

3.4.1 Noise Inconsistencies

Mahdian and Saic [41] present a blind method to detect splicing from an image to the other by detecting the noise inconsistencies in the falsified image. The authors first perform a one-level wavelet analysis of the image and then divide the image into a grid to estimate the noise block per block. The authors use white Gaussian noise model. Finally, they merge blocks with similar noise estimation and generate a set

(a) Input image.

(b) image with “legal” modifications.

(c) image with strong forgery and virtual noise.

Figure 3.7: 4.2(a): The input image used for the tests. 3.7(b): The input image with some “legal” transformations such resize, color enhancement, contrast, ... 3.7(c) Input image with a strong forgery. This image has been denoised and renoised with virtual noise. *Courtesy of Michel Couprie and Warren Miconi.*

of partitions with homogenous noise levels. Pan et al. [42] perform a similar image partition using the kurtosis values of natural images in band-pass filtered domains.

3.4.2 The device sensor fingerprint

The device sensor fingerprint is a sensor pattern noise that can be extracted from the PRNU of a set of images from the same camera. Chen et al. [43], Fridrich [44] and many others like [45, 46] use these fingerprints for device identification. Experiments

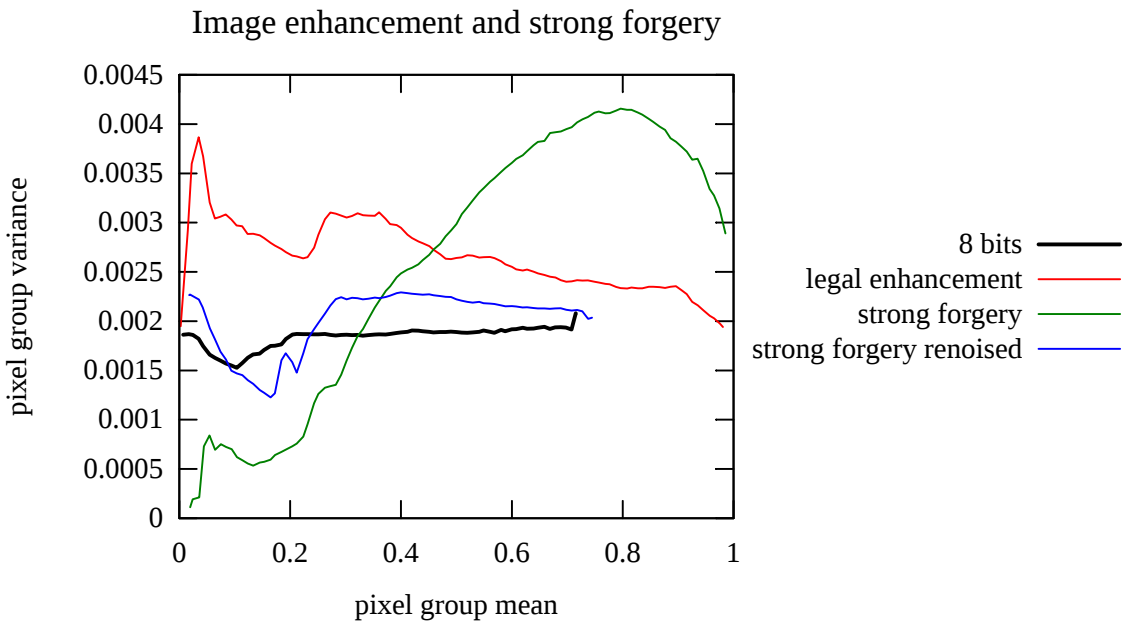


Figure 3.8: Image noise curves for "legal" vs. strong image manipulation and for a renoised strong forgery (with an additional contrast modification).

reported in [43] show promising results even for JPEG compressed images down to a quality factor of 75.

3.4.3 Computer Graphics

Some image forgeries can include some Computer Graphics (CG) parts when splicing is not possible. These CG image areas may have unusual noise, or no noise. Dehnie et al. [47] look for traces of PRNU in the image and consider the areas with singular PRNU as forgeries.

3.4.4 Color forgery

Hou et aL. [48] detect hue modification by analysing the correlation of the PRNU from each color channel.

3.5 Adding Artificial Noise

Adding artificial noise can serve several purposes: used on artificial images, it can help to test noise models and algorithms. On natural images, it can be used as a kind of filter, to imprint an image with an old-fashioned grainy feel, or, sometimes,

to camouflage an alteration. For our tests, we use the C++11 random number distributions to generate our Poisson and Gaussian distributions.

3.5.1 Artificial Images

Artificial images are ideal to test noise addition models, as they come free of any noise. We have used them to test and confirm the noise model proposed in Eq. (4.1) for raw images. They also allow to check the consistency of the noise model throughout various intensities and bit depth.

3.5.2 Natural Images

In the case of noise addition in natural images, the questions depend on the objective. If the noise addition is for a purely esthetical value, then the simple addition of a white Gaussian noise is enough. However, if the purpose is to cover other alterations, then a few parallel considerations have to be given thought. First, it is necessary to simulate some FPN, especially if there are several images coming from a single source being altered. Second, the added noise has to be coherent with the type of the image. The type of noise will be different, Gaussian for an 8-bit image and Poisson-Gaussian for a 16 bits one. In this second case, the noised value at pixel s is obtained with Q and N from Eq. (4.1):

$$I_{noised_s} = \alpha Q_s\left(\frac{1}{\alpha} * I\right) + N_s$$

The last thing to consider is the necessity or not to denoise the image before adding noise. In the case of a splicing, for example, it is necessary to denoise beforehand: indeed, the spliced section may have different noise characteristics than the rest of the image, adding overall noise won't camouflage the difference. A preliminary denoising will help reduce this discrepancy.

3.6 Noise and Anti-Forensics

The objective of this section is to point out some forensics methods that fails to detect digital image forgeries if some artificial noise is added on the falsified images. Indeed, adding artificial noise may affect forgery detection method dealing with

noise (Section 3.4) but also some methods where the pixel values distribution are important. On the following tests, artificial noise is added following the indications of section 3.5, with a very light Gaussian noise with $\sigma = 2.5$, meaning that about 30% of the pixels are not modified, due to quantization.

3.6.0.1 Double JPEG Compression

The double JPEG detection method introduced by Popescu and Farid [49] is based on an effect of the quantization step of the JPEG compression. Adding some noise on the falsified image will remove the quantization artefact and thus strongly decreases the double JPEG detection rate. Figure 3.9 shows the Fourier analysis of the first DCT coefficient for an image saved in JPEG, then saved again and finally artificially noised and saved in JPEG. The double JPEG artefact are much reduced, and so considerably more difficult to detect.

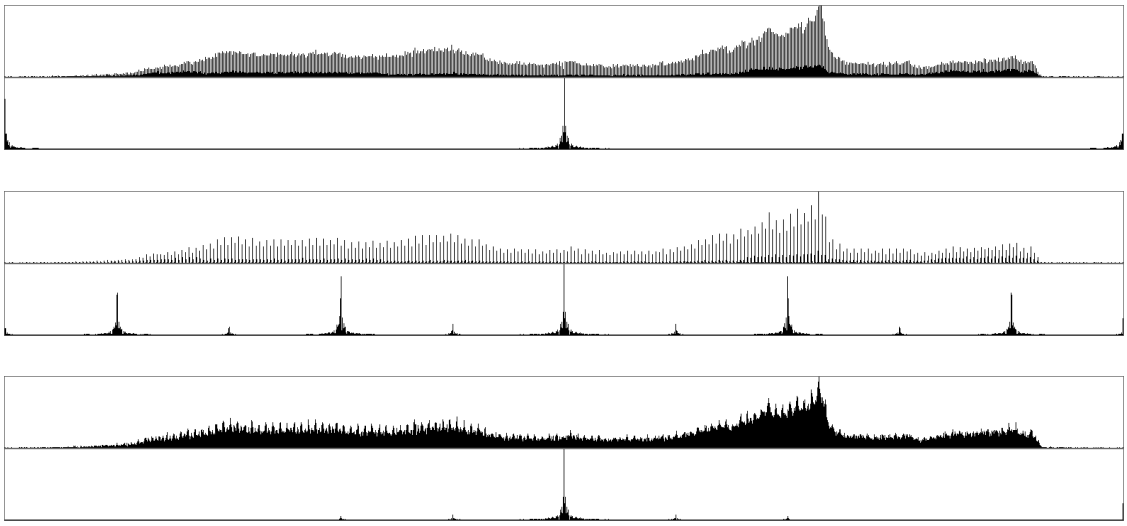


Figure 3.9: These graphics correspond to single JPEG (top), double JPEG (middle), and double JPEG + artificial noise + JPEG (bottom). For each image, the upper part is the histogram of the first DCT coefficient and the lower part to its Fourier transform, where the peaks reveal a double JPEG.

3.6.0.2 JPEG Ghost

The jpeg Ghost method presented by Farid [24] is an extension of the double jpeg that handle local properties of the image. Surprisingly, the method is not altered by noise, unless it is implausibly high. Figure 3.10 shows the result of this

method on a random image with the middle part previously saved in another jpeg quality than the overall image. Note that Stamm et al. [50] successfully disguise the JPEG compression history of an image by adding noise directly on its JPEG DCT coefficients. However, this process leaves slight image alterations that can be detected by [51]. Some recent methods can overcome this alterations issue [52].

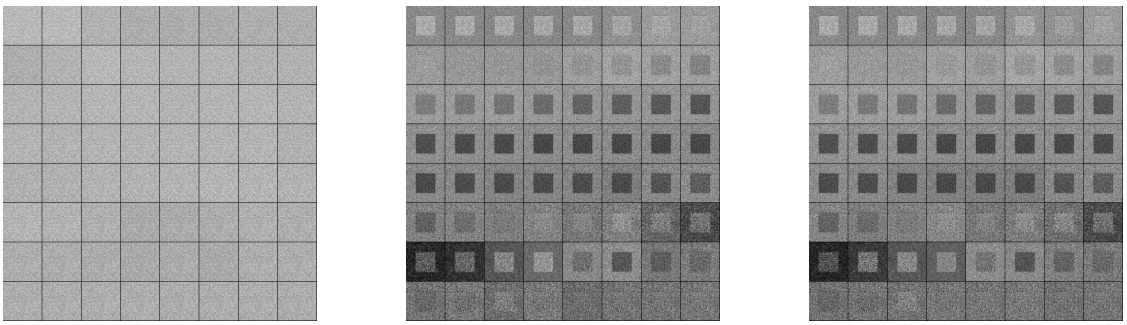


Figure 3.10: JPEG ghost [24]: (Left) 64 levels on a random image. (Middle) the 64 levels on the random image where the middle square part is previously saved in another jpeg quality than the overall image. (Right) Same as middle, but with artificial noise before the last JPEG saving.

3.6.0.3 Histogram based methods

Some histogram-based methods are used to detect global pixel intensity modifications, typically from Lookup-tables (LUT). Stamm et al. [25] detects the residual peaks of the image histogram resulting from such transformation, by analysing the frequency spectrum of the histogram. Adding some noise on the modified image will strongly affect the LUT modification, as depicted in Figure 3.11.

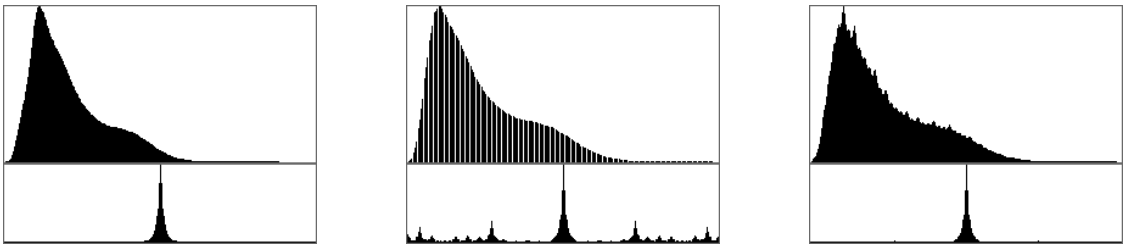


Figure 3.11: These graphics correspond to an original image (left), a modified image by applying a LUT, here contrast and brightness (middle), and the middle image with artificial noise (right). For each graphic, the upper part is the histogram of green component of the image and the lower part to its Fourier transform, where the peaks reveal the LUT operation.

3.7 Conclusion

In this chapter, we have detailed the various sources and models of noise in digital images. Then we have explored a large panel of noise alterations, related to both the acquisition pipeline and the post-processing. We have shown how these alterations affect the quality and intensity of the noise, and study the precise impact of each of those alterations. A major observation we have made is that, by the time we get to a JPEG image, even a high-quality one, the noise is extremely different from its original form in the raw image and is strongly affected by the successive image processing operations. From a statistics point of view, it seems extremely challenging to use this final noise for forgeries detection. Finally, we have looked at the consequences for image forensics and anti-forensics based on noise analysis.

千里之行，始於足下

A journey of a thousand miles begins with a single step

— Laozi's *Tao Te Ching*

4

Splicing Detection based on noise characteristics

Contents

4.1 Introduction

In this chapter, we will present three different methods to detect splicing in digital images by exploiting the noise statistics. Sections 4.2 and 4.3 will focus on the detection in raw images. Section 4.4 will extend our range of application to JPEG images. The main contribution of this thesis, the contribution histograns, will be presented in Section 4.3 and expanded in Section 4.4.

4.2 Splicing Detection in raw images based on noise characteristics

4.2.1 Introduction

In this section, we will introduce a simple noise-based approach to splicing detection using image noise, in raw images. This method is based on separating the image in tiles, extracting the characteristics of the noise in each tile, and then repeating the process with smaller tiles. The tiles are then separated in two categories using a PCA transform in order to look for strong outsiders.

The focus on raw images gives us access to an information that one wouldn't find in JPEG images: the Poisson component of the noise, as explained in Sec. 3.2.2. This allows us to do our tile classification using two characteristics, which gives us more robustness. Additionally, the noise in raw images is not corrupted and permits fine differentiation between tiles.

4.2.2 Digital Image Noise estimation

In order to obtain our noise parameters, we first denoise the original image, using the BM3D method proposed by Dabov et al. [36], as said in Sec. 3.2.3. The Poisson and Gaussian noise parameters are estimated from the difference image between the denoised image by BM3D and the original noised image. We first divide the

pixel luminance range into n equal intervals $\mathcal{I}_i$, $i \in [1, n]$. The pixels of the denoised image with intensity in $\mathcal{I}_i$ are grouped together to compute a variance σ_i of these pixels in the noise image and a mean value m_i in the denoised image. As specified by [32], the noise that appear in the lower and higher pixel intensities is not reliable. Thus, these pixels are discarded from our noise estimation process. The plot of the variance as a function of the mean gives a line that can be fitted with RANSAC line fitting. The line slope corresponds to the Poisson noise parameters whereas the "y-intercept" corresponds to the Gaussian noise component. An example of this fitting is depicted in Figure 4.1. For numerical robustness purpose, pixel groups with only a few pixels are discarded. In the rare case where a line have negative parameters, either Poisson or Gaussian, this line is discarded and not taken into account for the rest of the process. This can occur when the denoising is suboptimal or when the line fitting fails.

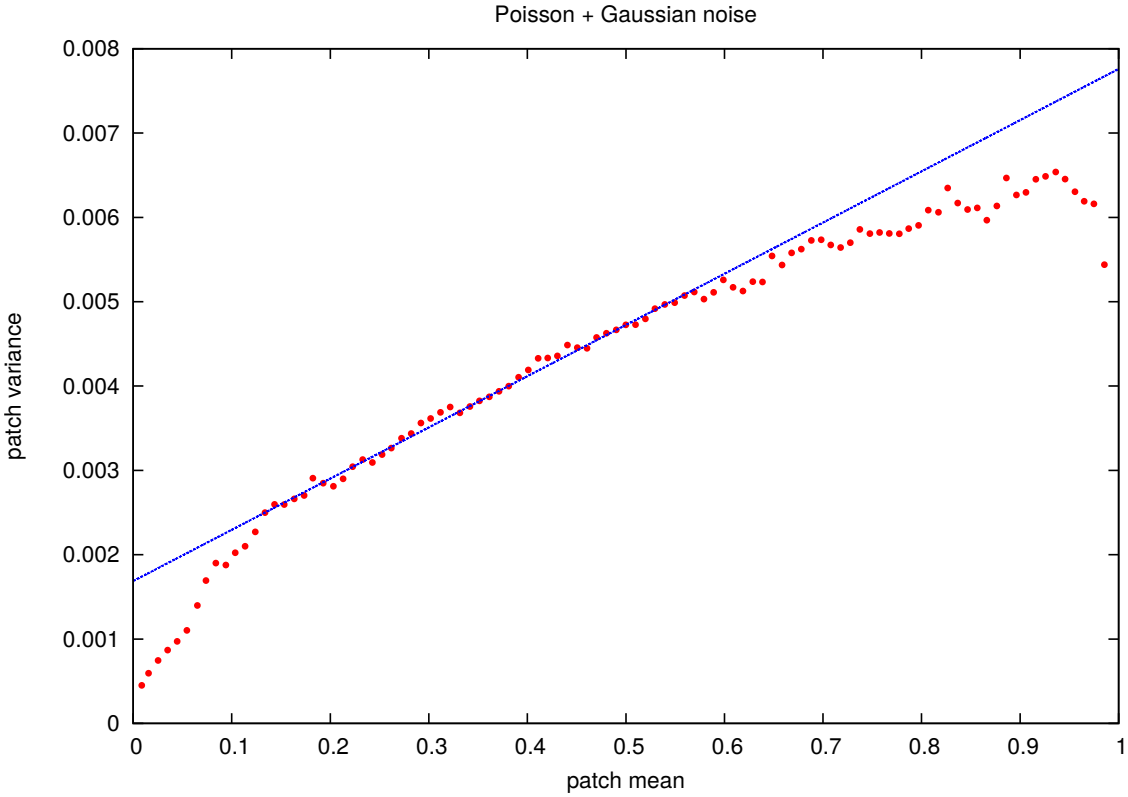
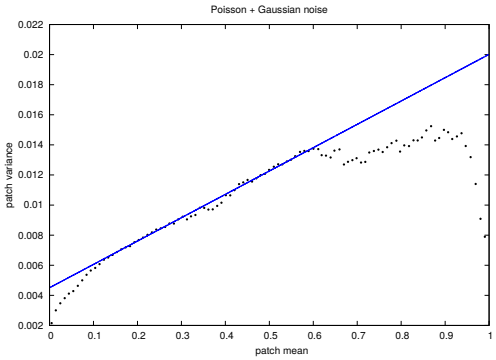
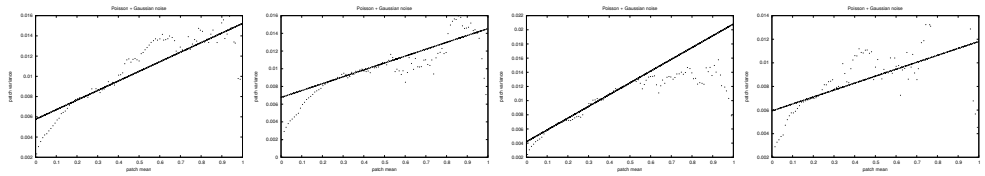


Figure 4.1: Patches variances according to the patch mean. The lower and higher pixel intensities are discarded. The line is fitted with RANSAC.

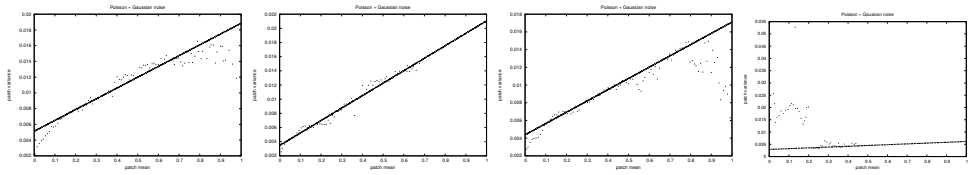
(a) Input image



(b) full image noise estimation



(c) Top-left quarter (d) Top-right quarter (e) Bottom-left quarter (f) Bottom-right quarter



(g) Some 1/16 of the image (h) Some 1/16 of the image (i) Some 1/16 of the image (j) The spliced zone

Figure 4.2: 4.2(a): a spliced image. The grid represents the quadtree sub-images. 4.2(b): Line-fitting of the noise data for the full image. From 4.2(c) to 4.2(i): Line-fitting of the noise data for various 1/16 subimages. 4.2(i) Line-fitting of the noise data for the spliced 1/16 sub-image.

4.2.3 Block based approach

The purpose of this method is to detect a splicing from the difference of noise between the two images (or more) used in the forgery. Our approach consists in

the computation of a quad-tree of the image regions. The statistics of the falsified region may differ from those of the rest of the image. This section describes how to build this quadtree data and how to detect the falsified zones.

4.2.3.1 Image block noise estimation

A significant constraint of BM3D and most of the denoising methods is the requirement of a good noise estimation of the input image before processing. Indeed, the denoising strength will be chosen according to the estimated noise in the image to process. Thus, a falsified region of the image may not be denoised properly during the denoising process.

In practice, the falsified region can be correctly denoised when the image is processed block per block, according to the respective proportion of original and falsified part in the block. In that case, the Poisson and Gaussian component of the falsified part can be estimated properly and compared to the rest of the image.

In this purpose, our method uses a quadtree to decompose the falsified image into sub regions. The quadtree approach is motivated by the noise statistic computation constraints. Indeed, very small regions provide an accurate spatial information of the forgery coordinates but may not have a wide enough variety of pixel intensity to compute the Poisson and Gaussian noise parameters. On the other hand, large image region can provide robust noise statistics but are not precise enough for forgery localization. It is noticeable that a small non falsified block of the image will not be denoised in exactly the same way it is denoised when the process is performed on the full image. A sample image with some subimages noise fitting is shown in Figure 4.2.

4.2.3.2 Gaussian-Poisson space

The quadtree processed on the falsified image generates a set of image regions. Usually, 3 levels of quadtree, that generate a total of 21 images, are enough. Each image is used to estimate a Poisson and a Gaussian noise component. The blocks corresponding to non altered parts of the image will provide similar noise parameters whereas the region corresponding to the splicing area may provide different noise parameters.

We use a clustering method on the Gaussian-Poisson space to identify the potential suspicious regions. This Gaussian-Poisson space express each noise estimation in term of coordinates (x =*gaussian*, y =*poisson*). Similar noise should have similar position. An example of a usual point distribution is shown in Figure 4.7.

At this step, the point distribution is discriminative enough to visually identify the splicing area but the automatic detection is not straightforward. The next section describes how to identify an outlier.

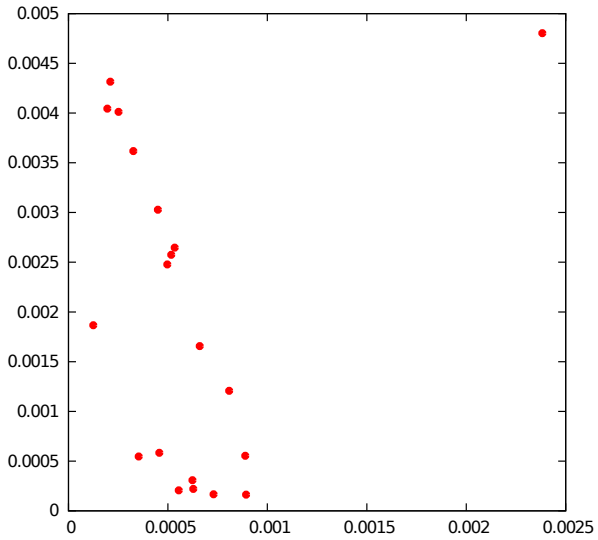


Figure 4.3: Each point represents the noise of a sub-region of the image. The horizontal axis represents the Gaussian standard deviation of the subimage and the vertical axis the Poisson standard deviation. The isolated point on the top right corner corresponds to the splicing area, its noise parameters are different from the parameters of the rest of the image.

4.2.3.3 Clustering

The point cloud obtained from the Gaussian-Poisson space should be transformed so that the outliers can be detected easily. Ideally, the point cloud of the non-altered sub-image should be grouped in a more compact form. A Principal Component Analysis (PCA) would perform well in a data set without outliers, but in our situation, an outlier located far from the main point group will alter the PCA transformation, due to the L_2-norm sensitivity of the PCA computation.

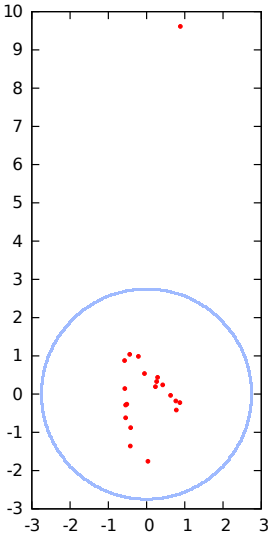


Figure 4.4: After the robust Principal Component Analysis, the point cloud corresponding to the non-falsified sub-images are grouped in a disc shape. The outlier, in the top can be detected with a σ-clipping.

4.2.3.3.1 Robust Principal Component Analysis To perform a robust PCA, we first remove a large set of potential outliers candidates. A point is considered as an outlier if the distance to its nearest neighbor is the highest of the group. To make our method robust to 2 splicings from the same image, with their representation in the Gaussian-Poisson space with two points near from each other, we better select an outlier as a point whose distance to its second nearest neighbor is the highest of the group.

Then our process removes not only one, but many isolated points with an iterative form of this method. Indeed, the PCA will still perform well if we remove much more points than the expected number of outliers.

Finally, the data normalization along the axis of the PCA space is computed only from the remaining points. Then this normalization is apply to all the data points, including the potential outliers. The initial point distribution that initially roughly looks like an ellipse should finally look like a compact group of points, except for the outliers, as depicted in Figure 4.4.

4.2.3.3.2 Outlier detection From the compact form of the data described in the previous section, the outliers can easily be extracted by a σ-clipping: all

points with distance to the origin higher that twice the average distance from the origin is considered as suspicious.

Moreover, for robustness purpose, the average distance from the origin σ is computed without the 3 points whose distance from the origin is the higher.

Finally, we use a progressive σ-clipping. Data points with a distance to the center of the disc shorter than two σ are considered unaltered. From two σ onward, the probability of a patch being altered increases linearly, until it reaches one when the most distant point or four σ, whichever is greater, is reached. This allows us to evaluate the impact of false positive: while datapoints that are not representative of an altered patch can be found outside of the two-σ boundary, they are often fairly close to it, and thus can be disregarded in comparison with more distant, and thus suspect, points. The boundary values have been chosen empirically, and are the ones that give the best results.

4.2.4 Results

4.2.4.1 Implementation

The raw images are loaded using LibRaw [38]. We could test our method on various raw image format files, like DNG, RAW, NEF, etc. The denoising process is performed by BM3D using the Matlab code provided by [36]. We empirically chose 100 value intervals to extract the noise statistics in section 4.2.2, inducing a line fitting over a maximum of 100 points.

4.2.4.2 Experimentations

We conducted our tests on real data with an automatic process. The database consists in 400 raw images with splicing, and 20 non spliced images to test the robustness of the method against false positive. The spliced images were generated automatically by copying random areas from other images in a random 1/16 of the test image. We conduced the tests on the green channel of every raw image.

On spliced images, we detect a splicing in 84% of the cases, and in 70.6% of those detection the localisation of the splicing is correctly identified. On unspliced images, we have a rate of false positive of 30%. A sample splicing detection is shown

on Figure 4.5. This method, however, has a few weaknesses: first of all, on the size of the spliced areas. An area that is too small will not affect much the noise characteristics of the subimage containing it, and thus will not be detectable. It is also impossible for this method to detect spliced areas with noise characteristics similar to the ones of the image being altered. We have also tested the method in the case of two simultaneous splicing from two different images, with success, see Figure 4.6. The discrimination method adopted for the PCA offer an important robustness in the case of multiple splicings. We conducted some tests on JPEG files. The results are promising but not convincing yet because of the lossy compression characteristic of JPEG. Indeed, the method seems to measure a kind of JPEG noise rather than the Gaussian and Poisson noise considered in this section.

4.2.5 Conclusion

We present a fully automated method to detect splicing in raw digital images. This approach is based on the relative consistency of noise parameters throughout an unaltered image. By looking for inconsistencies in those noise parameters in a quad-tree decomposition of the image, which are representative of both the Gaussian and Poisson components of the noise, we show that it is possible to highlight spliced areas in an image. The method is fully automatic, based on a robust PCA of the noise parameters and a subsequent σ-clipping, which detects the potential image splicings, and returns an image showing the probability of falsification by patch.
The main drawback of the method, in this state, is its lack of accuracy: indeed, a falsification that is too small will not have a sufficient impact on its tile to trigger the differentiation. In the same way, a big splicing equally divided between four or more tiles would skew the results. The solution to this second problem would be to use sliding tiles and make a composite result image, but the time cost of the denoising step would become very problematic.

4.3 Splicing Detection using the Contribution Histogram

4.3.1 Introduction

In this section, we present a more complex method to detect splicing in raw images, using the image noise. For this purpose, we introduce a new tool: the noise density contribution histogram, based on an analysis of the image noise probability density function. Once again, our method starts with a denoising and tiling step, followed by the construction of noise density contribution histograms for each tile. It is by comparing those histograms pair-wise that we can classify them in order to detect the splicing.

4.3.2 Noise Density Contribution Histograms

4.3.2.1 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 Figure 4.7. The exact function defining the Poisson-Gauss probability density table is shown in Eq.4.1, where σ is the standard deviation of the Gaussian portion of the function and α a scaling parameter applied to the Poisson portion:

$$f(v_d, v_n) = \frac{\alpha}{\sigma\sqrt{2\pi}} \sum_{x=0}^{\infty} \frac{(\alpha v_d)^{\alpha x} e^{-\alpha v_d}}{(\alpha x)!} \exp\left(\frac{-(v_n - x)^2}{2\sigma^2}\right) \tag{4.1}$$

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 Figure 4.7.

A cross-section of the table along a single denoised value follows a Poisson-Gauss probability distribution (see dark line in Figure 4.7). 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 (Figure 4.8). The objective of our method is to differentiate these two contributions, and to identify the parts of the image that participate in each contribution.

A naive approach would be to try to fit a model, such as the one in Eq.4.1, 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.

4.3.2.2 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 Figure 4.8, 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.

4.3.2.3 Classification between $\wedge$ type and $\vee$ 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. 4.9(a) and 4.9(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 $\wedge$ type. If there is a clear majority outside of this axis, it is a $\vee$ type (Fig. 4.9(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.4.10, middle column).

4.3.3 Seed Expansion

Once every block has been assigned a primary type (be it $\wedge$, $\vee$, 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 $\wedge$ or $\vee$. The similarity s between two contribution histograms $C_1(v_d, v_n)$ and $C_2(v_d, v_n)$ is simply a sum of term by term absolute difference, but only in

the rows where both histograms 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 $\wedge$ shape group, and thus a probability $1-p$ to belong to the $\vee$ shape group. These probabilities are computed with an iterative scheme with an initial value set to 0.5. Then, each undefined block probability is iteratively set as the weighted average probability of the N blocks whose contribution histogram 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. Finally, if a block probability is high enough, the block is considered as being fully in said category. The result can be seen in Fig.4.10, right column.

4.3.4 Multichannel

In order to improve our results, we apply the whole process - denoising, noise density histogram, contribution histogram, classification, and expansion - to each channel of the input image (R, G_1, B, G_2). Although the grey world assumption [62] 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 merged and averaged.

4.3.5 Implementation and Results

The raw images are loaded using LibRaw [38]. The denoising process is performed by BM3D using the Matlab code provided by [36]. 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 dependant on the quality of the denoising. Although the procedure we used [36] 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 4.1. 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 4.1: The detection rate on spliced and unspliced images.

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

4.3.6 Conclusion

In this section, 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. In the next section, we adapt this method for JPEG images.

4.4 Splicing Detection using the Down-Projection Contribution Histogram

4.4.1 Introduction

In this section, we present our third method to detect splicing in digital images. It is both an improvement over the method presented in the previous section, and an extension to the JPEG images, making it more efficient and giving it a wider range of application. This method exploits the statistical properties of the noise density function and is based on the assumption that the inherent noise in images is different enough from one image to the other that their density functions differ. We also presume that at least one of the images is otherwise unaltered.
We first address the subject of noise density histogram, and its transformation to a noise contribution histogram. The next part introduces the concept of noise contribution down-projection, which is the core of our method. The following section shows how to use this new tool to detect splicing, accompanied by several pre- and post-processing steps. Finally, we expose our results and compare them to other state-of-the-art methods.

4.4.1.1 Denoising

The first step of our method is to denoise the image. In section 4.3, following a similar approach, we used the denoising algorithm proposed by Dabov et al. [36]. Although extremely effective on RAW images, this denoiser is designed for 16-bit

images and relies on a Poisson-Gaussian model for the noise. As our method is to be applied on 8-bit JPEG images, and since in chapter 3 we have shown that the noise of those images can be considered as a zero-mean Additive White Gaussian Noise (AWGN), we consider two state-of-the art denoisers: the non-local means method implemented by Buades et al. [34], and the BM3D implementation by Lebrun [63]. Although both methods give comparatively similar results, the approach by Buades et al. is faster, so we use that one in our experiments.

4.4.2 Noise Density Function Models

As explained in chapter 3, the noise in a single image can be assumed to follow a single noise probability density function. As such, it is expected that noise in a spliced image can be divided in two parts: the noise from the original image, and the noise from the spliced element, as shown in Fig. 4.11. Although the overall noise probability density function should not be altered much if the size of the spliced element is small compared to that of the image, a local analysis of this function may allow a differentiation between the two parts.

4.4.2.1 Noise density histogram

The noise density histogram of a discrete signal is a representation of the probability density function of the noise contained in the signal. In our method, we build the noise density histogram based on the original image I_O and its denoised version I_D. As such, each pixel location p of the image is assigned a pair of values, $v_n(p)$ and $v_d(p)$, corresponding to its original noised value and denoised value respectively. Then, the value of the noise density histogram H at any point (i, j) is the number of pixels respecting the condition $(v_n(p), v_d(p)) = (i, j)$ or, more formally:

$$H(i,j) = \operatorname{card} P_{i,j}, \;\; P_{i,j} = \{p \mid v_n(p) = i \text{ and } v_d(p) = j\}$$

However, for the histogram to be representative of a probability density function, each column j (denoised value) needs to be be normalized with a normalizing constant C_j, where $C_j = \operatorname{card} p \mid v_d(p) = j$.

If we assume an image with an equipartition of values, i.e. each possible intensity is represented by an equal number of pixels, and add a perfect AWGN of standard deviation σ, then the noise density histogram can be modeled as the basic Gaussian probability density shown in Fig.4.12. In this situation, we expect the value of the normalized histogram to be roughly equal to the equivalent Gaussian probability density:

$$H(i,j) \simeq \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(i-j)^2}{2\sigma^2}}$$

Fig. 4.13 depicts the noise density histogram computed on a real JPEG image, from Algo. 1. It can be seen that the comportment of the extremities of the histogram differs from the rest, which is due to the saturation of the noise. In our experiments, we ignore those two extremes.

Algorithm 1: Global density histogram generation

Input: original image: I_O
denoised image: I_D
Output: global density histogram: H

```
// init the histogram
H.fill(0)
// fill the histogram
for p ∈ I_O do
    Increment(H(I_O(p), I_D(p)))
// normalize the histogram
foreach column c of H do
    normalize(c) such ∑_i H(c, i) = 1
```

4.4.2.2 Noise contribution histogram

A noise contribution histogram represents the contribution of a subimage to the noise density histogram of the whole image. A value at any point in the histogram can be interpreted as the number of pixels fulfilling the values condition of this point contained in the subimage, divided by the number of pixels fulfilling this same condition contained in the full image. As such, it is an element-wise division of

the noise density histogram of the subimage by the global noise density histogram, both non-normalized. More formally:

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

The contribution histogram values are upper-bounded by 1, indicating that all the pixels couple values (v_d, v_n) are contained in the subimage. The sum of the contribution histograms of non-overlapping subimages covering the entire image will be an histogram with a shape resembling that of the global noise density histogram (i.e. a gaussian noise density function), with the values all equal to 1.
The advantage of the contribution histograms is that they allow to distinguish noises variations, even when they are small. Assuming the noise is always centered, a small change in standard deviation (<1) will not alter the width of the distribution, but will change its overall shape, allowing differentiation as shown in Fig. 4.14. An example of the noise contribution histogram of a subimage is shown in Fig. 4.15.

4.4.3 Noise Contribution Down-Projection

One of the main issues with the noise contribution histograms is that their intensity distribution corresponds to the one of the subimage, e.g. a subimage with a majority of dark pixels will have an histogram with values concentrated in the lower rows and columns of the histogram. Consequently, it is hard to compare two subimages with highly different intensity ranges, since their contribution histograms will have little or no intersection. In chapter 4.3, there is no real way around that, since the noise follows a Poisson-Gaussian law and as such is inherently spatially variant. However, when dealing with pure AWGN, we know that the noise distribution is independent of the base intensity. As such, if we first deskew the histogram as shown in Fig.4.16, so that it is centered around a single line instead of around the diagonal identity, then histogram rows may be interchanged freely.

The concept of noise contribution down-projection derives from this independance. If rows can be interchanged, then it is also possible to exchange single histogram bins between rows, if they have the same signed difference between

their noised and denoised values, e.g. exchanging the point of coordinates (51, 2) with the one in (57, 2).

The construction of a noise contribution down-projection from a noise contribution histogram is as follows: first, the histogram is deskewed so that the identity axis is along a single value O, by default the 0 axis, using this equation:

$$\forall t \in H \begin{cases} H_R(t).x = H(t).x \\ H_R(t).y = H(t).x - H(t).y \end{cases}$$

where H_R is the skewed histogram and $.x$ and $.y$ respectively represent the first and second coordinates of a point in the histogram. Then, for each column, we remove all bins with a content of 0 and move down the upper elements of one histogram cell. The length of a column is then only the number of non-zero bins originally in it, as can be seen in Fig. 4.17 and applied to real data in Fig. 4.18.

The main justification of this data manipulation is that all of the separate contribution down-projections will have roughly the same appearance: the first column will be the largest, and they will get progressively smaller as we progress through. As a consequence, we will always compare the largest possible columns of a contribution down-projection to its average contribution equivalent, as depicted in Fig. 4.19.

4.4.4 Application to Splicing Detection

4.4.4.1 Tiling

In order to analyze the noise in an image I_O, we first denoise it, obtaining the denoised image I_D. As suggested in Section 4.4.3, these images are then divided into non-overlapping square subimages of identical sizes, referred to hereafter as tiles. The size of the tiles is an important consideration: we need enough pixels in a tile so that their contribution down-projections are statistically significant, but we also know that smaller tiles will yield a better resolution to pinpoint the location of the falsification. According to our tests, a size of 64×64 pixels provides enough statistical data, while giving a reasonable accuracy on medium to wide sized images.

4.4.4.2 Average Contribution down-projection

We then build the overall noise histogram, and the noise down-projection of each tile, using the method presented in Sec. 4.4.2. These down-projections are what will allow us to classify our subimages. In addition, we also construct an overall average contribution histogram (and its corresponding down-projection), by averaging all the non-zero contributions for each point of the histogram:

$$DP_{av}(v_d, v_n) = \frac{\sum_i^{Ns} DP_i(v_d, v_n)}{card\Big(\{DP_i) \mid DP_i(v_d, v_n) \neq 0\}\Big)}$$

where N_S is the number of subimages. Using all the contribution for this histogram (including the zero ones) would result in an flat histogram where each point has a value of $1/N_S$.

4.4.4.3 Classification of Sub-Images

For each tile, we compute the location of the maximal differences its contribution down-projection has compared to the average. Here, we consider only the lowest row of each down-projected histogram. It is expected that the tiles from the original image will have their maximal differences spread equitably around the central axis, while the tiles from the spliced element will have maximal differences either close to the axis or as far away from it as possible, depending on whether it has more or less noise than the original image, as seen on Fig.4.19. The score of a tile is then calculated based on those distances, and is obtained by taking the average distance between its highest computed differences and the axis.

4.4.4.4 Results Visualization Enhancements

The result given by the first process of classification, although readily usable, can be improved with several post-processing methods. The first thing that can be done is to apply a scaling on the results, while removing the current extreme values. This allows the classification to extend on a wider range, while avoiding to be misled by potential aberrant, outlier results. This normalized image is then submitted to a double threshold, whose values were found experimentally to give the best average

Algorithm 2: Image Classification

Input: set of contribution down-projections of each subimage: DP_i
average contribution down-projection: DP_{av}

Output: score of each tile: *score*

foreach DP_i **do**
  $score_i = 0$
  `// n: number of locations used`
  for $k = 0$ **to** $n - 1$ **do**
    $(maxDiff, maxLoc) = \mathbf{max}\Big(DP_i(t) - DP_{av}(t),\ \forall t \in DP_i,\ DP_i(t) \neq 0\Big)$
    $DP_i(maxLoc) = 0$
    $score+ = abs(maxDiff.x)$
  $score_i = score_i/n$

results on our database, in order to improve its readability by a human observer, as shown in Fig. 4.20. This results in an easier way to highlight possible discrepancies on the input image by comparing it to the threshold image. Finally, the whole method (pre-processing, main algorithm, scaling, and thresholding) is applied on each color channel, and on the luminance channel for additional information.

4.4.4.5 Formal Algorithm

The overall pipeline is outlined in algorithm 3.

4.4.5 Preprocessing

Due to the imperfect nature of the denoiser used, several steps are undertaken to decrease the impact of the denoising process on the final classification.

The first step is to remove the traces of JPEG blocking in both the original and the denoised image after denoising. Indeed, the denoiser used tends to consider the borders of the JPEG blocks as slightly more noised than the rest of the image, and as such denoise them more strongly. Although the suppression of the borders decreases the resolution of the image by around 43%, the removal of this added "artificial" noise increases the precision of the classification.

Algorithm 3: Method Formal Algorithm

Input: original image: I_O
Output: result image: I_R

```
// cf. Sec. 4.4.2
I_D = denoise(I_O)
// cf. Algo. 1
H = densityHistogramGeneration(I_O, I_D)
// cf. Sec. 4.4.4.1
T = createTiles(I_O,I_D)
// compute the contribution for each map
foreach tile T do
    // cf. Sec 4.4.3
    DP_i = contributionDPGeneration(T_O, T_D)
// cf. Sec 4.4.4.2
DP_av = averageContributionGeneration(DP_i)
// cf. Algo. 2
classifyTiles(DP_i, DP_av)
```

The second step consists in working around the contours in the original image. The denoising process has a tendancy to perform poorly around contours, and as such a replacement/elimination process is undertaken, in function of the amount of contour contained in a tile. We detect the contours in the image by applying a Laplacian filter, and then dilate them to ensure they contain all of the poorly denoised areas. Then, we check the percentage of pixels belonging to a dilated contour in each tile. If the percentage is lower than a defined threshold, set to 60% in our experimentations, we replace the contour pixels in the tile by random non-contour pixels also belonging to the tile. However, if the percentage is too high, we simply flag the tile so that it is removed from consideration for all subsequent computations. A similar process is applied to saturated tiles, depending on their number of pixels with either too high (>250) or too low (<5) intensity.

Finally, in some cases the denoiser causes noise discrepancies that can appear even in non-spliced areas of the original image. This can be corrected by applying a linear amplification to the noise according to the image pixel intensity, with a minimal impact on the spliced element. This method is detailed in Algo. 4.

Algorithm 4: Noise preprocessing

Input: original image: I_O
noise image: I_N
Output: result noise image: I_{RN}

```
// global scale coefficient
```
$$k = \frac{\sum_{i,j} I_O(i,j).|I_N(i,j)|}{\sum_{i,j} \left(I_O(i,j).|I_N(i,j)| \right)^2}$$

```
// new noise image using I_O intensity
foreach pixel (i, j) do
```
$$I_{RN} = k.I_O(i,j).I_N(i,j)$$

4.4.6 Tests and Results

The method was tested on a set of 400 spliced images generated from 200 natural images from the Dresden dataset [64], with sizes between 2592×1944 and 3648×2736. This dataset was selected instead of the usual Columbia dataset that becomes outmoded, notably in term of image size and quality. The images used cover interior and exterior scenes, and were taken using different brands and models of cameras. The noise standard deviation range from 0.36 to 1.4. The algorithm runs in around 1 minute for the biggest images on consumer-grade hardware. The general results are are shown in Table 4.2.

Table 4.2: Average results in normal image condition

	F1 Score	MCC
Mahdian and Saic [41]	0.133	0.110
Zeng et al. [65]	0.078	0.098
He et al. [59]	0.139	0.120
Our Method	**0.203**	**0.178**

The results are computed using the F1 score and the Matthews Correlation Coefficient (MCC) [66]. Indeed, although the F1 score is more frequently employed, in our database the spliced element tends to be small compared to the overall size of the image, usually around 1/16th of the image. As such, the MCC seems to be more adapted than the F1 score which gives the same weight to both areas,

Table 4.3: Results with increased JPEG compression (using MCC)

	JPEG 95	JPEG 90	JPEG 85	JPEG 80
Mahdian and Saic [41]	0.110	0.068	0.068	0.073
Zeng et al. [65]	0.098	0.099	0.105	0.116
He et al. [59]	0.120	0.124	0.134	0.119
Our Method	**0.178**	**0.177**	**0.161**	**0.137**

regardless of their respective sizes. This scoring method ranges between -1 and 1, with zero being a random guess on each pixel. Additionally, we ran our algorithm on a dataset of 300 artificial images, with a perfect denoising. The noise conditions were set to replicate those of natural images (ie, a noise standard deviation between 0.36 and 1.4, and the spliced element having randomly more or less noise than the base image). The average MCC on this dataset was 0.41. This shows a significant impact of imperfect denoising on the overall efficiency of our method. As such, it is reasonable to expect an improvement on detection capabilities as denoising methods will improve.

Table 4.3 also shows that our approach is more impacted by a reduction in image quality than other state-of-the art methods, although our results still remain better than the alternatives. Finally, Table 4.4 shows that our method is also more resistant to strong downsampling.

The two most important factors to explain the robustness of our approach against compression and downsampling are the comparison to the global histogram and the post-processing enhancements. As compression and downsampling are applied equally across the whole image, the differentiation between the original pixels and the spliced is still possible, though the difference is attenuated. Using the global histogram down-projection allows us to reduce the impact that strong effects would have on small patches of the image. Although the final difference in scores is reduced in the result images, the normalization and thresholding applied afterwards enhances even the small variations in score, facilitating the final discrimination. This robustness can be seen as a consequence of the ability of our method to discriminate even on low-noise images.

Table 4.4: Results with downsampling (using MCC)

	Normal conditions	Downsample 75%	Downsample 50%
Mahdian and Saic [41]	0.110	0.093	0.010
Zeng et al. [65]	0.098	0.099	0.102
He et al. [59]	0.120	0.095	0.072
Our Method	**0.178**	**0.157**	**0.140**

4.4.7 Conclusion

In this section, we present a new approach to detect splicing in JPEG images, based on the analysis of the noise density throughout the image. We have shown that our method performs better than other state of the art approaches on a dataset extracted from the Dresden image database.

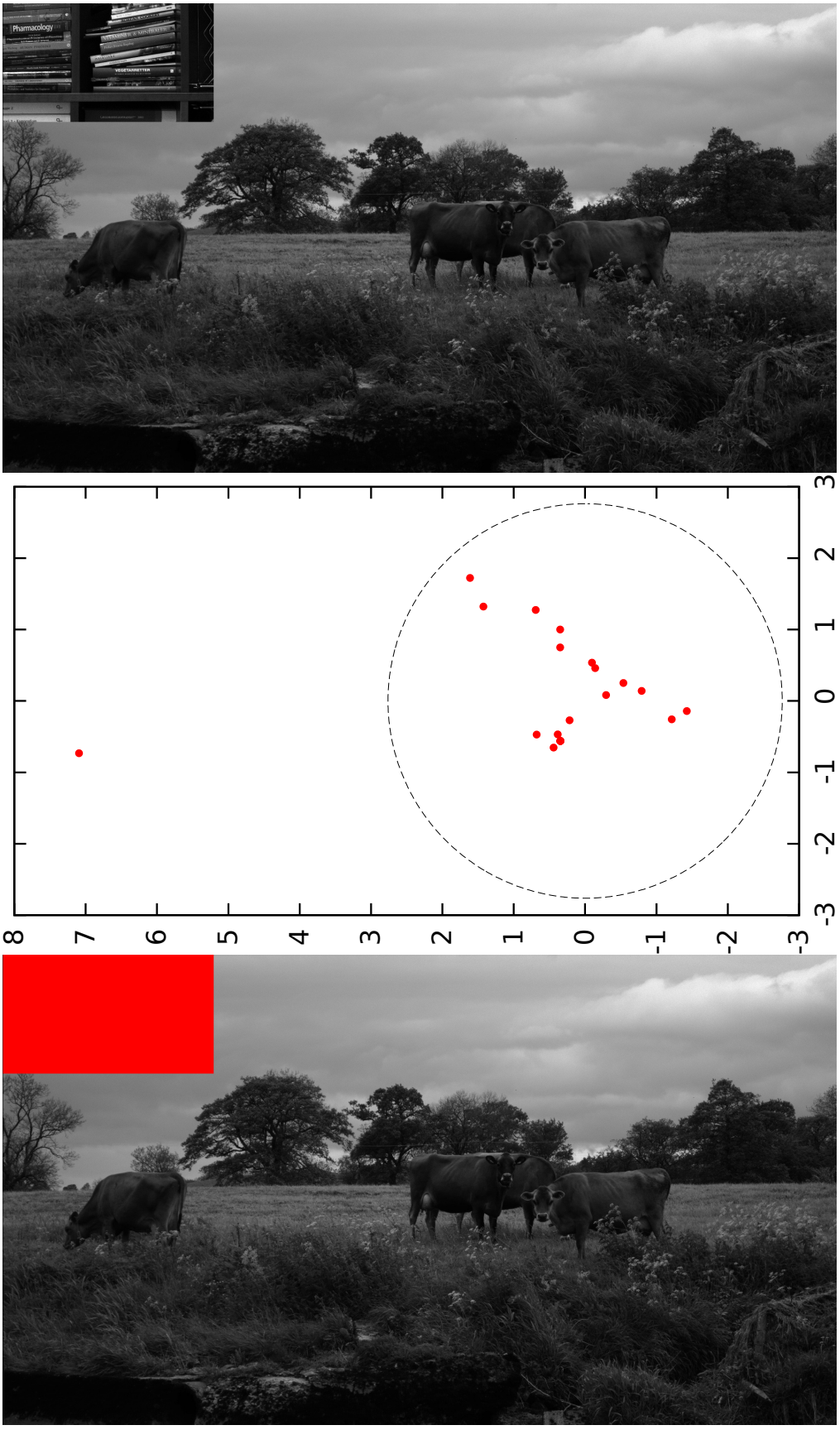


Figure 4.5: Top: a spliced image. Middle: the robust PCA and the σ-clipping. Bottom: the spliced zone detection (in red).

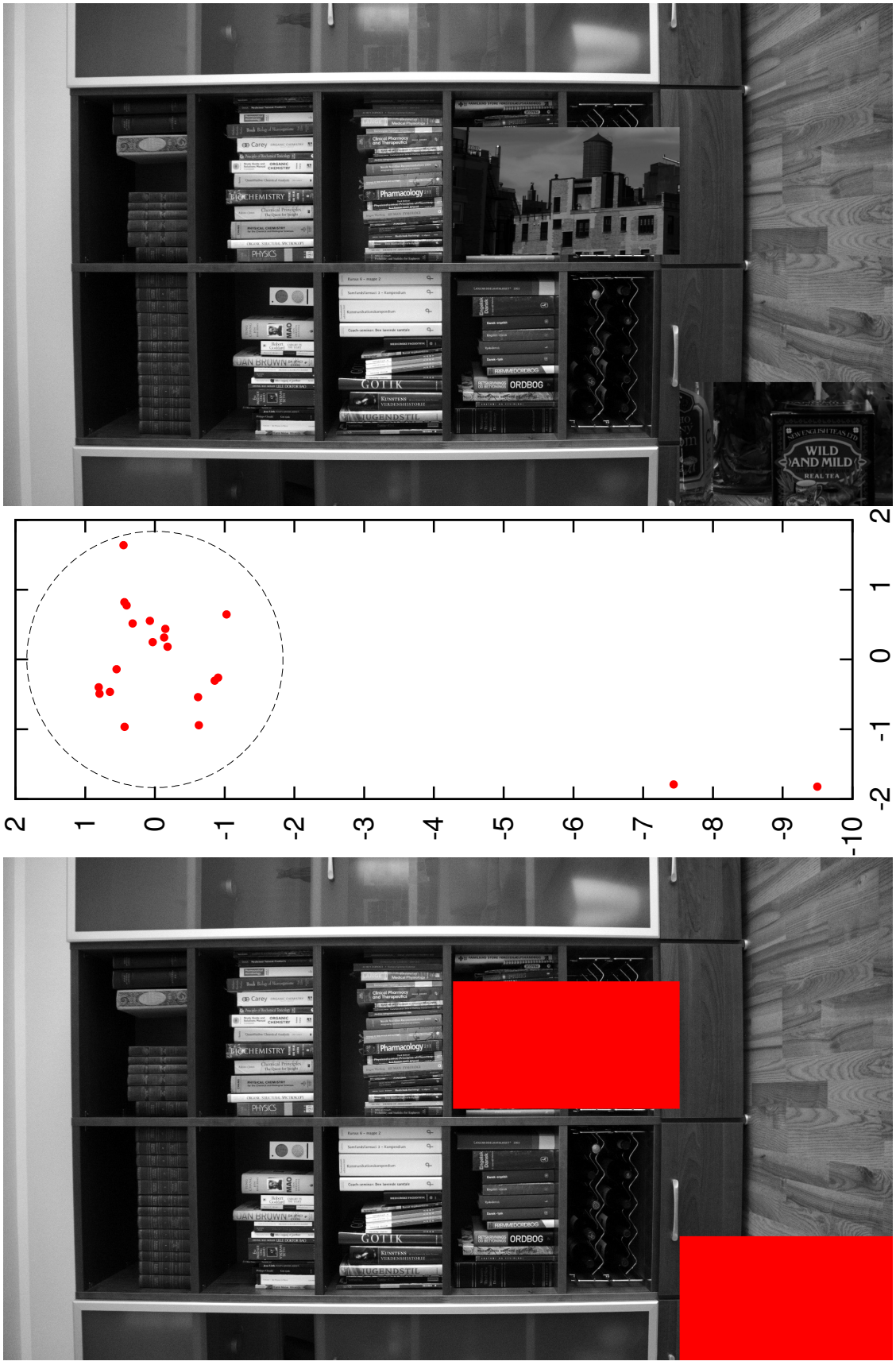


Figure 4.6: Top: a double spliced image. Middle: the robust PCA and the σ-clipping. Bottom: the spliced zones detection (in red).

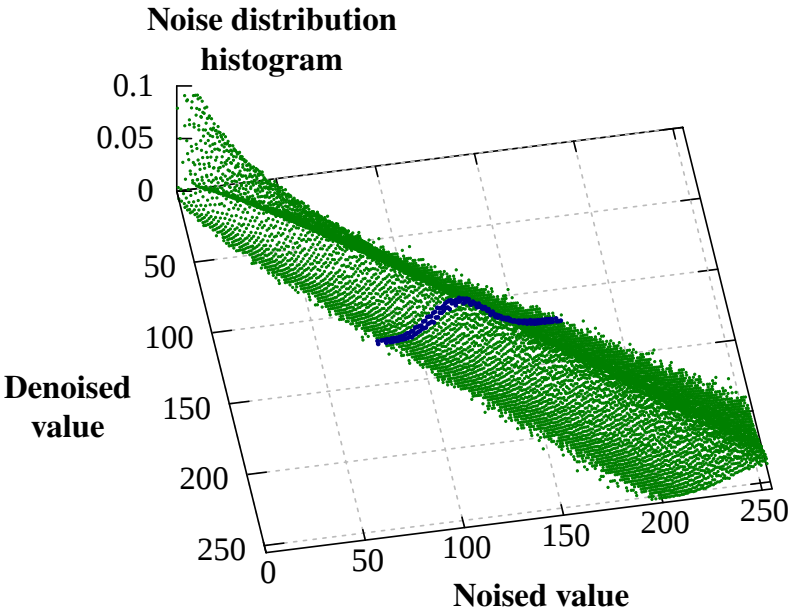


Figure 4.7: Poisson-Gauss density histogram. The dark line is a cross-section along a single denoised value.

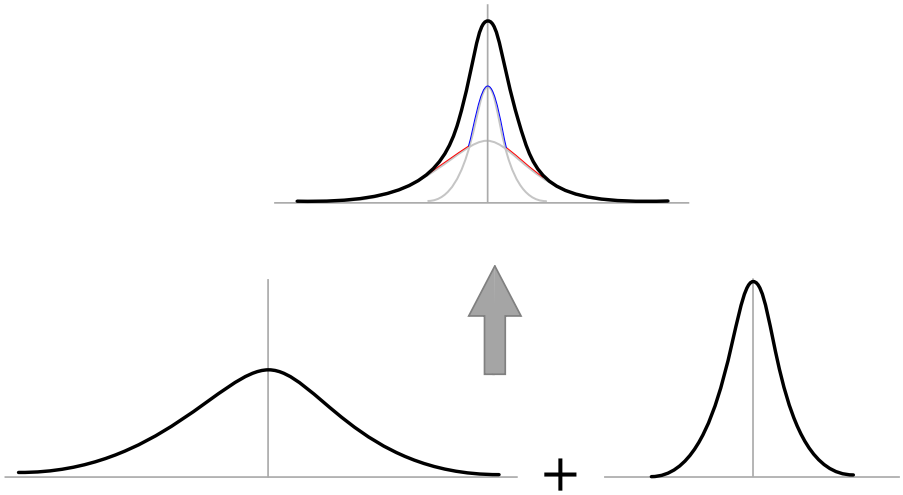
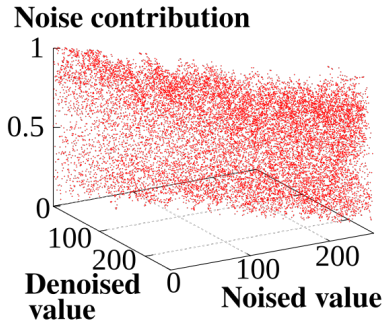
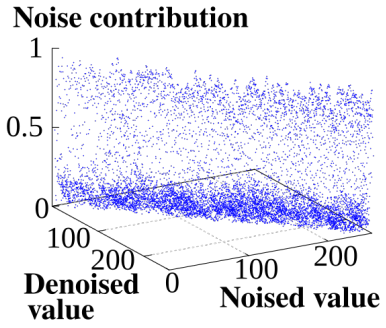


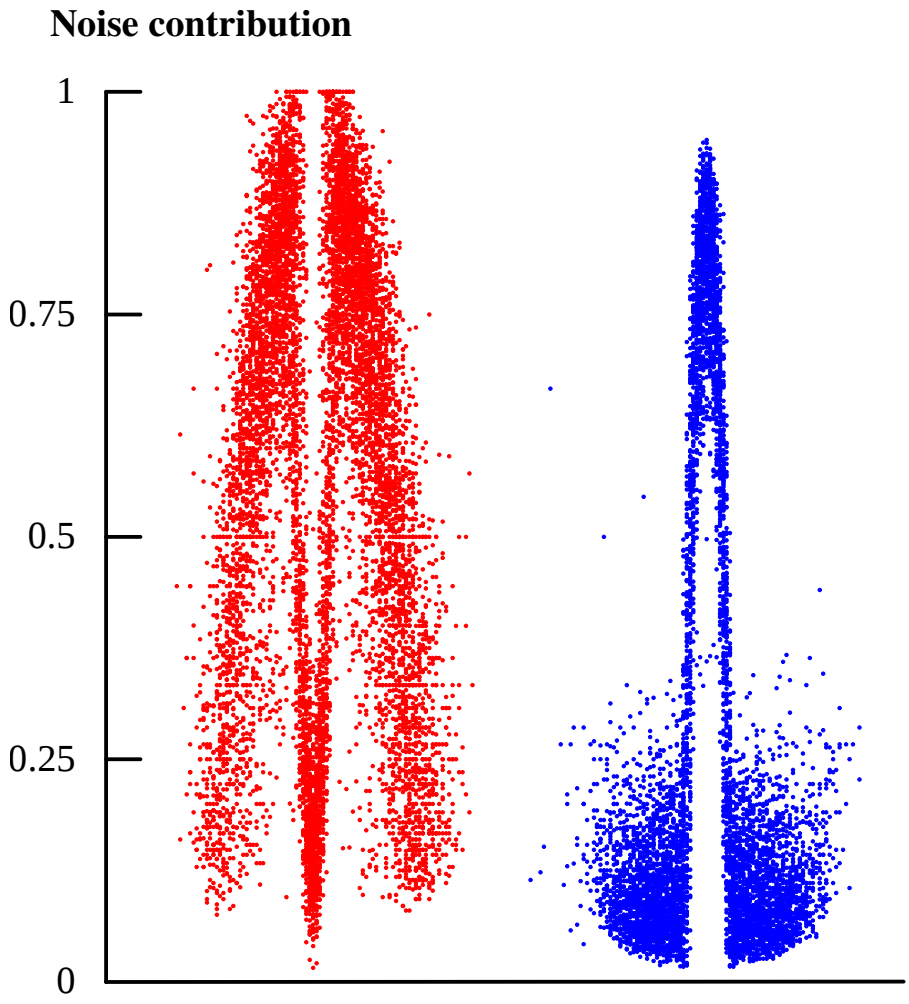
Figure 4.8: Density histogram is an addition of two curves.



(a) ∨ contribution table

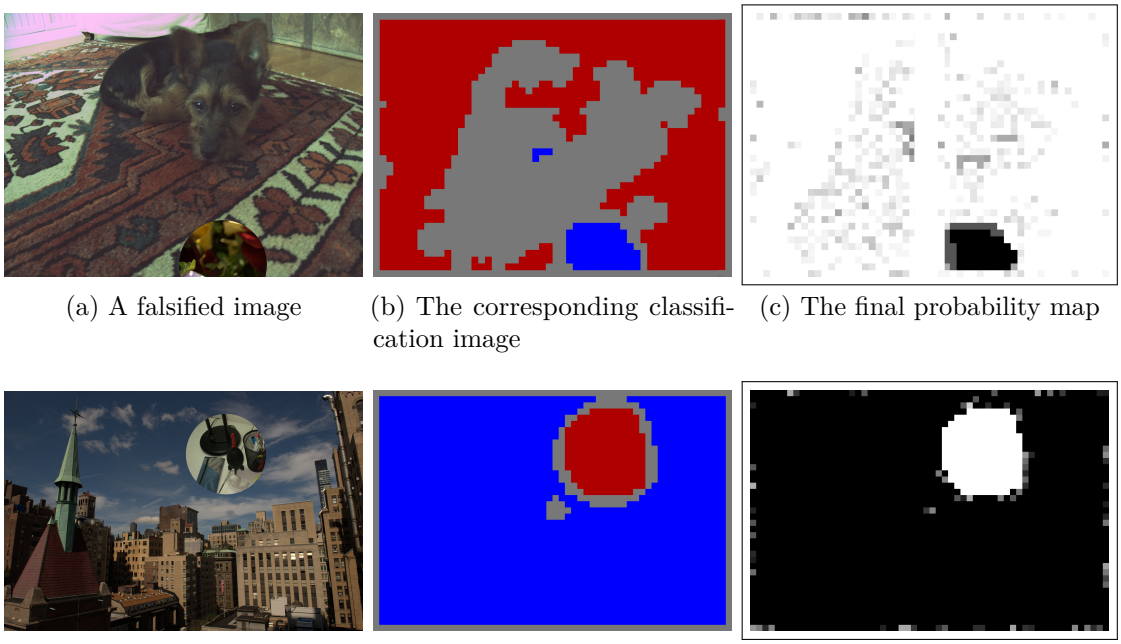


(b) ∧ contribution table



(c) Projections of each contribution table

Figure 4.9: The two types of contribution tables. 4.9(a) and 4.9(b) show their overall appearance, 4.9(c) shows their projection on a plane orthogonal to the identity axis.



(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

Figure 4.10: 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.

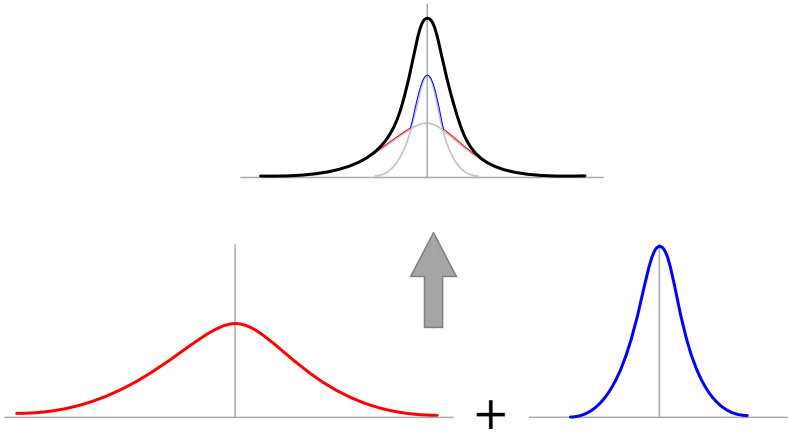


Figure 4.11: The noise density function in a spliced image remains Gaussian-like, and is the sum of two Gaussian functions.

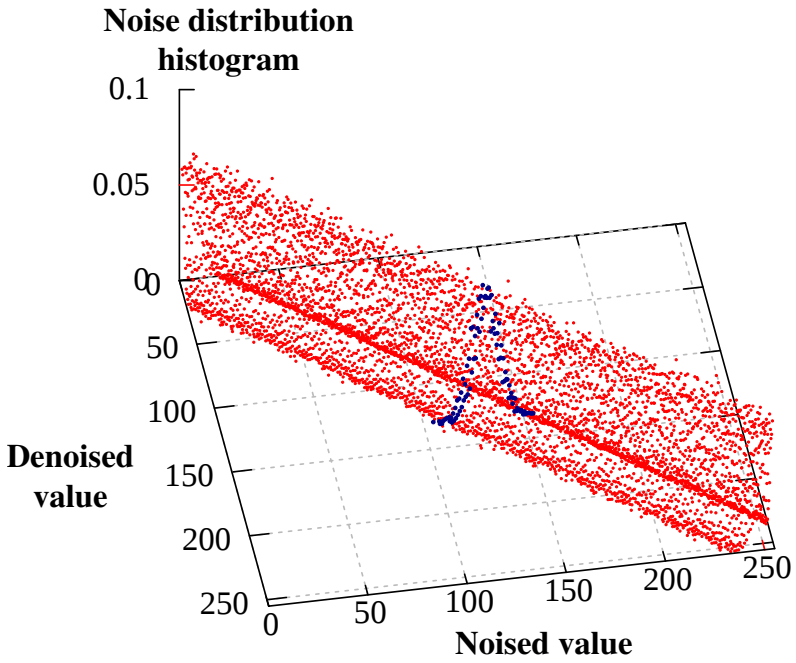


Figure 4.12: Ideal Gaussian noise density histogram. The dark line is a cross-section along a single denoised value, and represents a 1D Gaussian distribution.

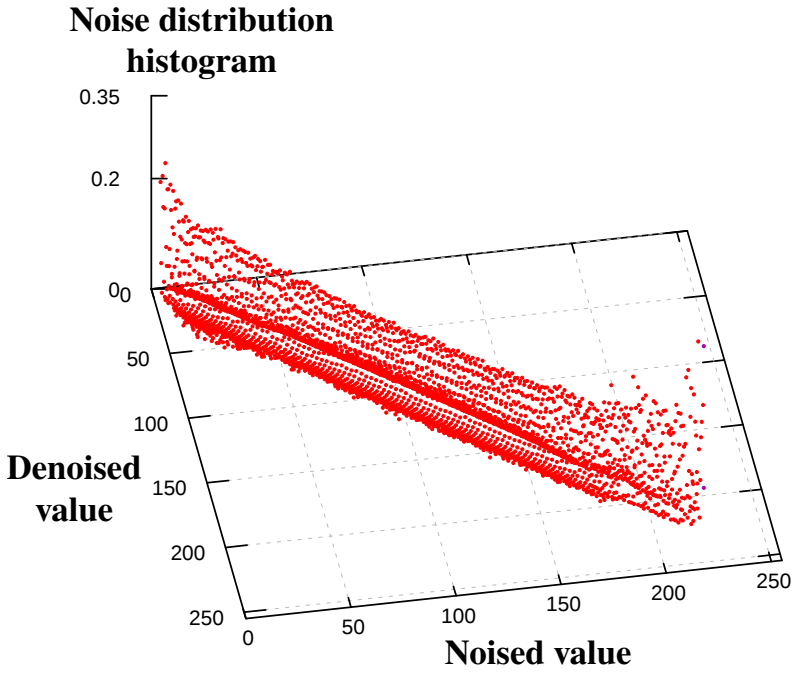


Figure 4.13: Gaussian noise density histogram obtained on a natural image with our algorithm.

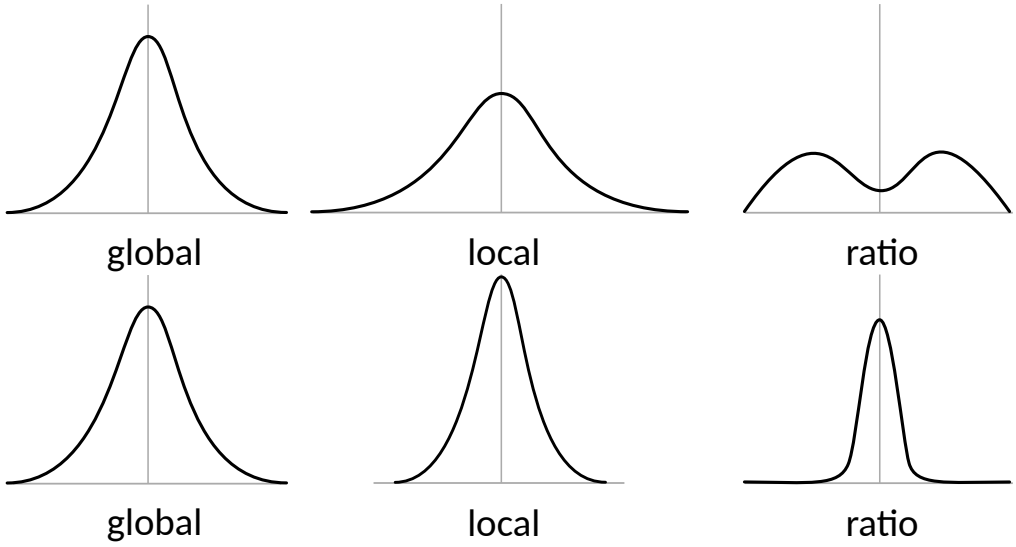


Figure 4.14: Ideal shapes of various contribution histograms. Top: spliced part with a higher standard deviation noise. Bottom: spliced part with a lower standard deviation noise.

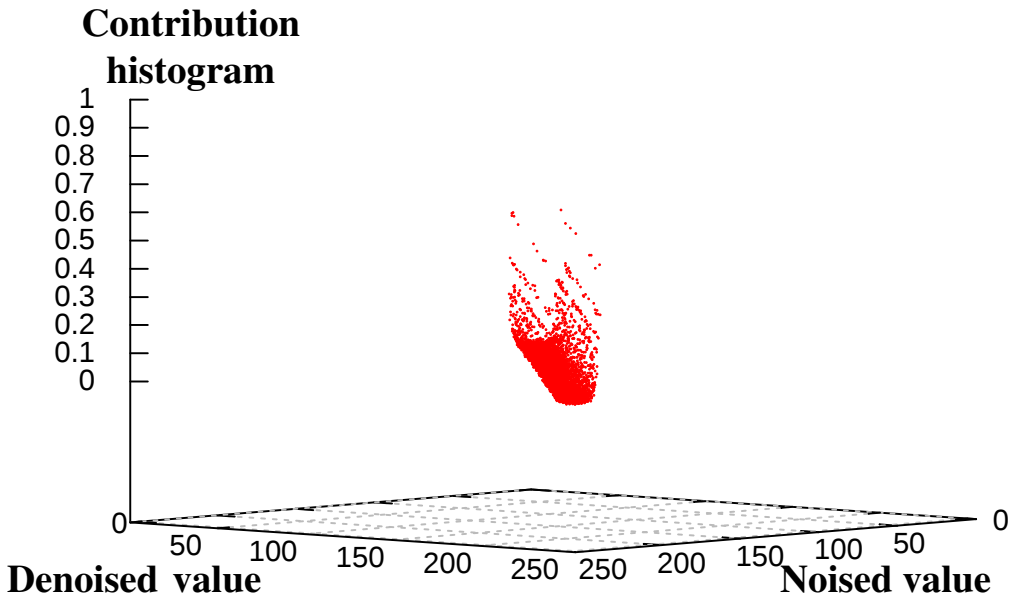


Figure 4.15: The noise contribution histogram of a subimage. The higher contributions on higher noises is normal, and is useful for identifying the spliced area.

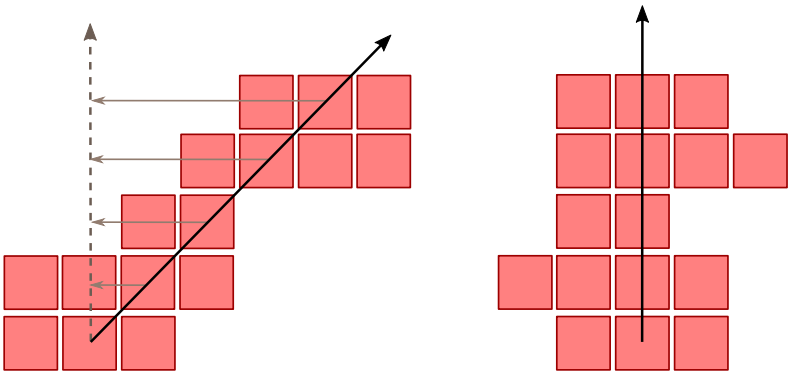


Figure 4.16: Deskewing the histogram. This is the first step before applying the down-projection.

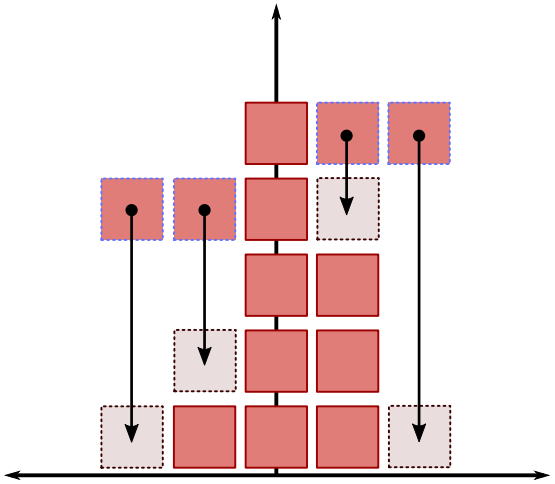


Figure 4.17: The process of down-projection.

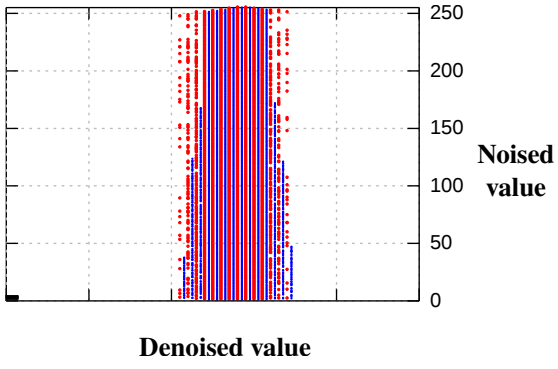
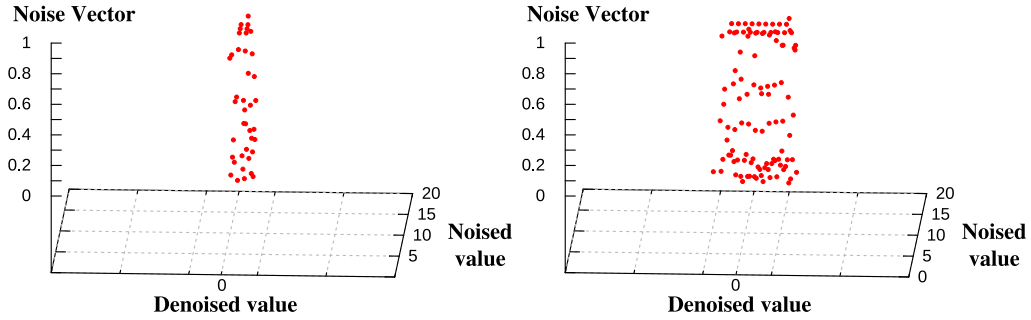


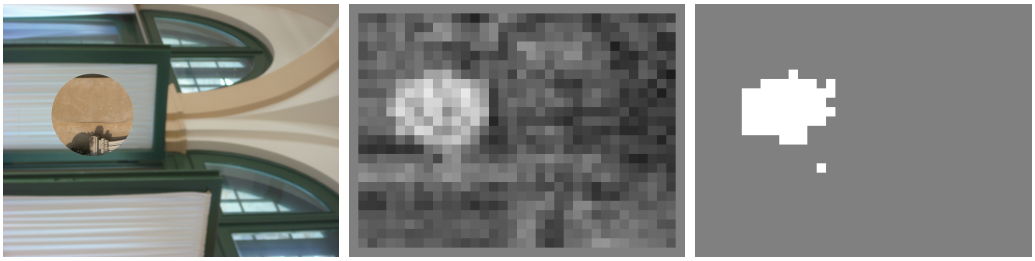
Figure 4.18: The skewed histogram and its down-projection side by side. We see that the down-projection is dense compared to the initial histogram.



(a) Contribution down-projection for a tile of the original image.

(b) Contribution down-projection for a tile of the spliced element.

Figure 4.19: The difference in contribution down-projections between a tile in the original image and one in the spliced element. We can see that the contributions of the spliced elementare much farther from the central axis on average.



(a) Spliced image

(b) Scaled result

(c) Thresholded result

Figure 4.20: An exemple of splicing detection.

Figure 4.21: Results comparison. 4.21(a), 4.21(b) and 4.21(c) are the spliced images. 4.21(d), 4.21(e) and 4.21(f) are our results. 4.21(g), 4.21(h) and 4.21(i) are Zeng et al. [65] results. 4.21(j), 4.21(k) and 4.21(l) are Mahdian and Saic [41] results. 4.21(m), 4.21(n) and 4.21(o) are He et al. [59] results.

A Photograph Is a Secret About a Secret. The More it Tells You the Less You Know.

— Diane Arbus

5 Counter Forensics

Contents

5.1 Introduction

Up to this point, all of the methods presented have been aimed at detecting falsifications, and more precisely splicing, in images. In this chapter, we will focus on a new, concurrent field, dedicated to countering the forensics methods: anti-forensics. The main idea behind anti-forensics is to find ways to hide the detectable signs of an alteration, without degrading its quality.

Another face of digital imaging are artificial images generated using computer graphics, referred to hereafter as CG images. The quality of CG images has drastically increased over the past few years, to the point that some of them can fool a human observer. Although they are commonly used in video games or movies, CG images can have more ill-intentioned uses. They are, for example, used in cases concerning identity theft, usually combined with a voice-alteration software. Although high-quality CG images can deceive the human eye, there are some physical or statistical properties that can be used by forensic methods to differentiate them from natural images. Consequently, as with natural images, some counter-forensics methods have emerged in order to hide those properties.

In this chapter, we present a novel way to transfer the noise from one image to another, and two different uses for it: first, conceal a spliced element in a natural image. Second, make a CG image appear more natural both from the human perspective and against forensics software.

5.2 Noise Density Transfer

A first naive approach to transfer noise from one image to another would be to denoise both images, obtain the noise image (difference between the original and denoised image) of the first one, and add it onto the second one. This, of course, causes several problems: the size of the two images has to be the same and the contours of the first image will be superimposed on the second one, due to the fact that all denoising methods leave traces correlated to the contours of the denoised image.

Another method would consist of estimating the noise variance in the first image and applying an artificial Gaussian noise with the same variance on the second image after denoising. Although this would produce a reasonably convincing result, at least for human observers, a perfect additive Gaussian white noise is never found in

natural images. Indeed, chapter 3 shows that though a standard digital image noise can be approximated by a Gaussian distribution, it is never perfect. Additionally, the denoising step will clearly leave some traces, which could be detected with adequate methods, such as the blur detection method presented in [67].

5.2.1 State of the art - Density Transfer

The probability density function associated to a random variable, such as noise in an image, describes the odds for this random variable to have a given value. The method we propose is based on transforming the noise density function of a first image so that it is as close as possible to that of another image. The problem of transferring and changing a density function has already been studied extensively in several optimization problems. The concept was first introduced by Monge in 1781 [68], and more recently studied statisticians in the 1970s [69] under the moniker of Wasserstein distances. It was applied to images by Peleg et al. in [70], where it is used to propose a method to change the resolution of an image while avoiding aliasing. Several optimized versions of the generic transfer function have been developed by Xu et al. [71] and Huang et al. [72]. However, the purpose of our approach is not only to match one probability distribution to another, but also to get a credible image noise as a result. Additionally, both of the optimized methods are used to process very large amounts of data, while in our case the quantity of changes to be done in a distribution are limited by the size of the image. As a consequence, we do not need to focus on the optimal number of operations to transform the distributions. The method may be implemented in a faster and more efficient way, but this would not affect the final result.

5.2.2 Noise transfer

The proposed approach is based on transferring the noise density histogram (introduced in Chapter 4.3 of an image onto another, in contrast to just generating a Gaussian noise with the correct standard deviation on the second image. The distinction between these two techniques is important: indeed, the proposed method can copy any kind of noise density histogram, and as such is not only limited to

images with Gaussian noise.

We note I_s the image source of the transfer and I_d the image destination of the transfer. The first step is to denoise these two images, giving us the denoised images $\widehat{I}_s$ and $\widehat{I}_d$. Note that these denoised images $\widehat{I}_s$ and $\widehat{I}_d$ are used only for the density histograms computation, but are not used in the noise transfer itself. Finally, we compute the corresponding density histograms T_s and T_d. See Fig. 5.1.

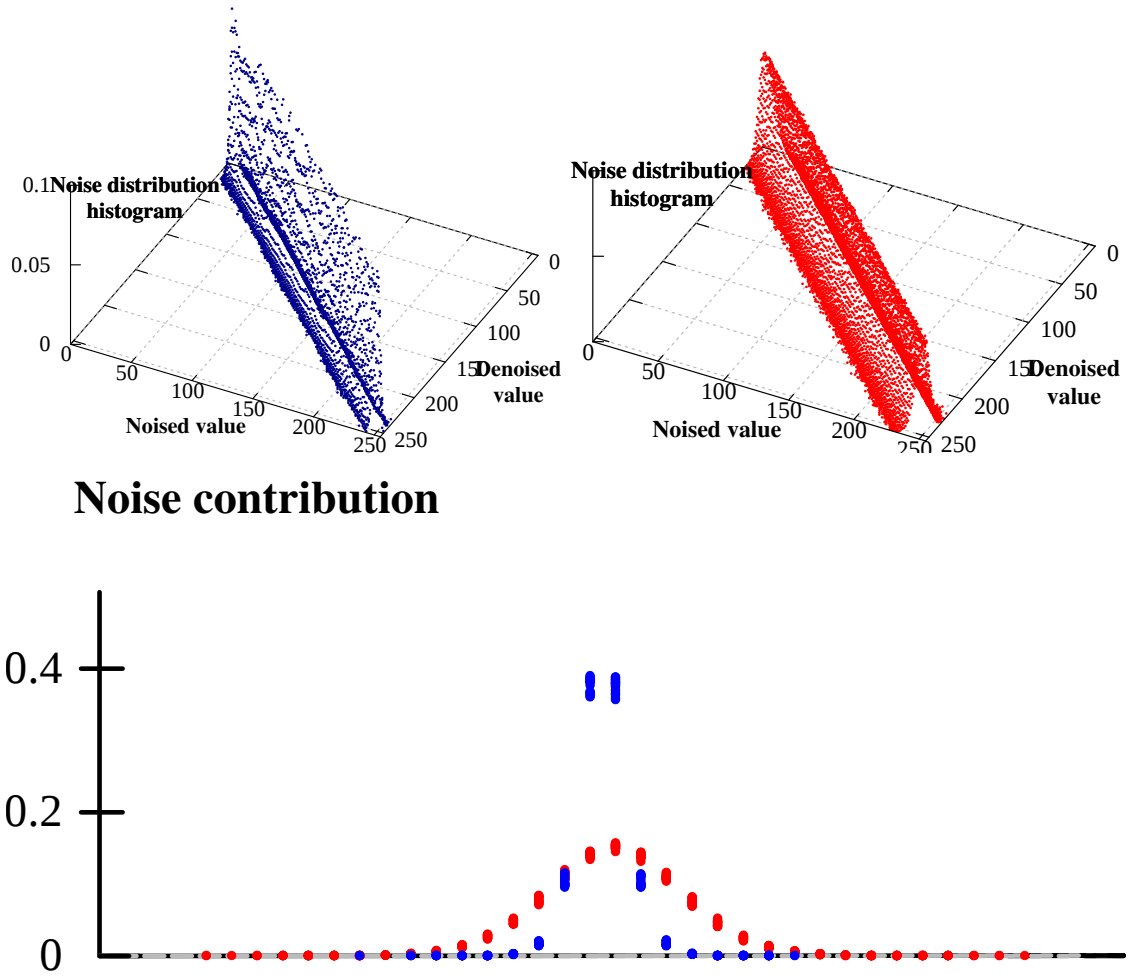


Figure 5.1: 5.1(a) and 5.1(b) are the density histograms of the source and destination images respectively. 5.1(c) shows a projection of those two histograms along the identity axis for easier comparison.

For the transfer, we consider that a density histogram is bounded as follow: each row represents the possible values of the denoised image, and each column

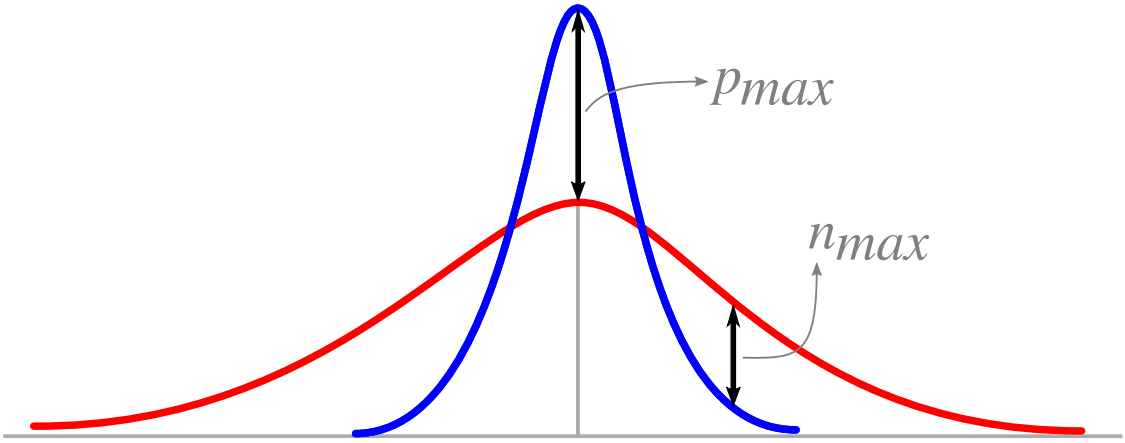


Figure 5.2: A representation of the maximal differences and their location. The curves represent a cut of the density function along a denoised value.

represents the possible values of the noised image. Both rows and columns can take a value ranging in $1...N$. For example, $N = 255$ for images with 8-bits per channel.

On a given row r, we look for the two maximal differences column indexes c_p and c_n. Theses indexes are formally defined as:

$$c_p | \Big(T_s(r, c_p) - T_d(r, c_p) \Big) = \max_{1 \leq i \leq N} \Big(T_s(r, i) - T_d(r, i) \Big)$$

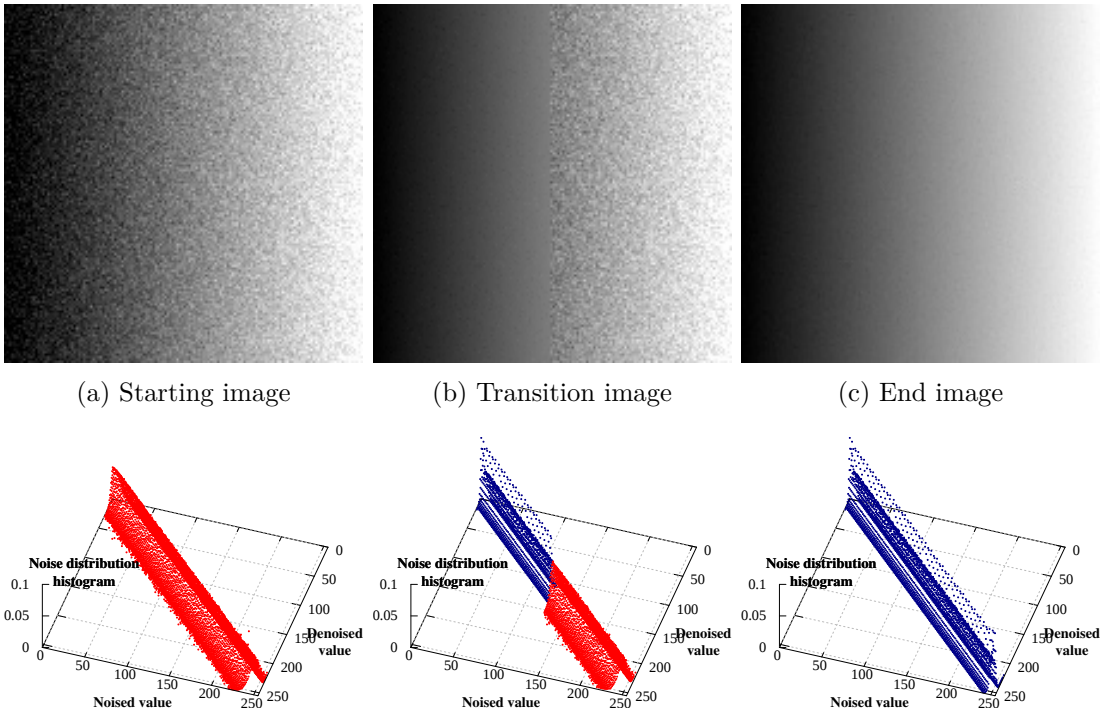
$$c_n | \Big(T_d(r, c_n) - T_s(r, c_n) \Big) = \max_{1 \leq j \leq N} \Big(T_d(r, j) - T_s(r, j) \Big)$$

We then define the corresponding maximal differences (see Fig.5.2) as:

$$p_{max} = \Big(T_s(r, c_p) - T_d(r, c_p) \Big)$$

$$n_{max} = \Big(T_d(r, c_n) - T_s(r, c_n) \Big)$$

Once the maximal differences and their positions are found, we change the value of a random pixel on the destination image with a noised value of c_n and a denoised value of r to a noised value of c_p. We then update the destination density histogram to reflect the change. This will have the effect of altering the shape of the density histogram T_d and bring it closer to T_s, by reducing both p_{max} and n_{max}. This process is iterated on the column until convergence. We consider convergence is reached when the destination density histogram takes the same configuration two



(a) Starting image (b) Transition image (c) End image

(d) Starting density histogram (e) Transition density histogram (f) End density histogram

Figure 5.3: Images in the top row show the impact of the noise transfer on an image. The noise has been voluntarily amplified so that the results are more easily visibles. The bottom row show the evolution of the density histogram through the transfer.

times in row. Once convergence is reached for a row, we process the next one until all rows are transferred, as shown in Fig. 5.3.

5.2.3 Gamut management

In the case where the destination histogram has a row which is empty on the source histogram, it is necessary to fill the row with coherent data for the sake of the transfer. In the case of a Gaussian distribution, we simply translate the closest non-empty row along the identity axis, as shown in Fig. 5.4. In the case of a Poisson-Gauss distribution, this may not work if the closest non-empty row is too far away and as such has a clearly different standard deviation. In that case, if we assume that the noise is independent in each channel, we can look in each of the other channels for a suitable replacement.

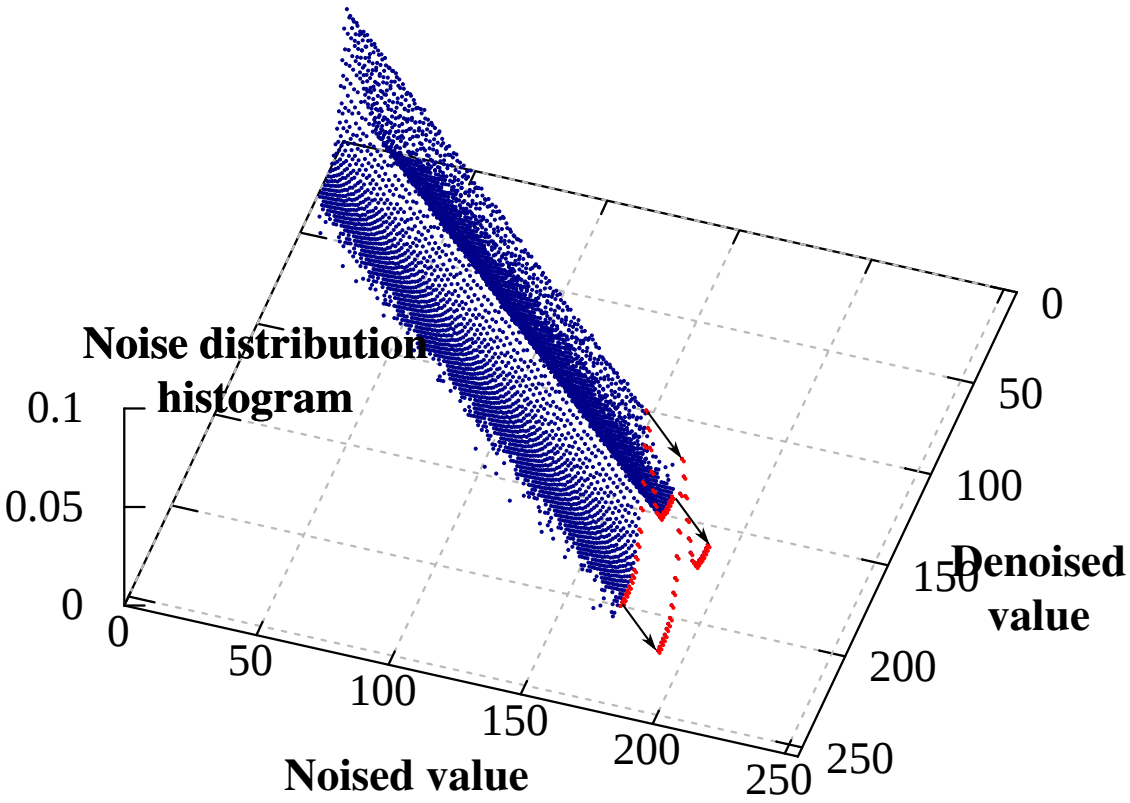
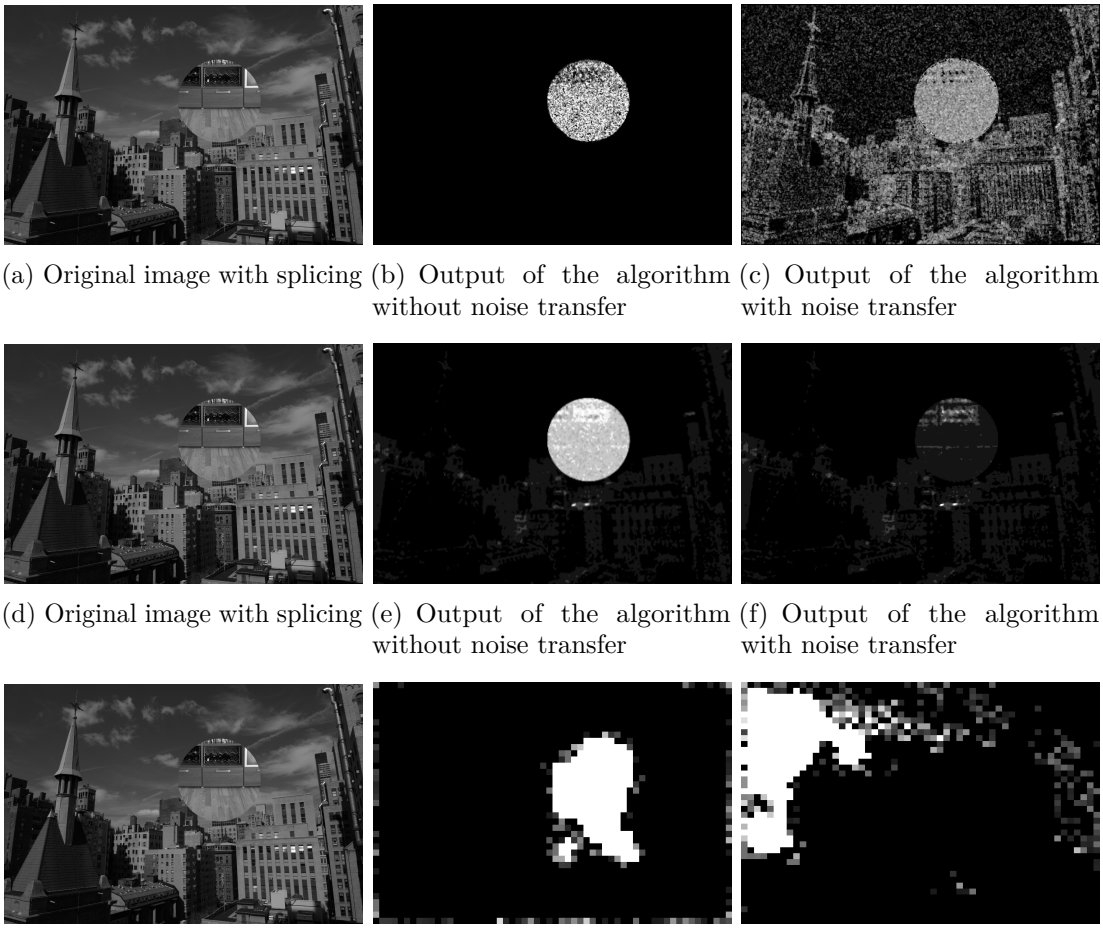


Figure 5.4: A representation of the gamut management. The red curve is translated and copied along the identity axis to fill the row where it is needed.

5.3 Application: splicing camouflage

5.3.1 State of the art

Several methods have emerged in recent years to counter the development of digital forensics. When it comes to conceal splicings, most of the anti-forensics approaches are focused on the JPEG format. In [73] and [17], the authors present counters to hide several alterations, like CFA disturbance or image resampling. Likewise, [74] propose to alter the transform coefficients of an image during compression, to hide the impact of several falsifications like copy-move, splicing, or double-JPEG on the compression history of an image. However, to our knowledge, there are no existing methods to dissimulate the impact of splicing on the noise in an image.



(a) Original image with splicing (b) Output of the algorithm without noise transfer (c) Output of the algorithm with noise transfer

(d) Original image with splicing (e) Output of the algorithm without noise transfer (f) Output of the algorithm with noise transfer

(g) Original image with splicing (h) Output of the algorithm without noise transfer (i) Output of the algorithm with noise transfer

Figure 5.5: The effect of our method on three detection approaches. 5.5(a), 5.5(b) and 5.5(c) show the effect on the approach designed by Pan et al. ; 5.5(d), 5.5(e) and 5.5(f) use the approach of Mahdian et al., and 5.5(g), 5.5(h) and 5.5(i) our approach from chapter 4.3. In all of the cases, the spliced element cannot be distinguished from the rest of the image after the noise transfer.

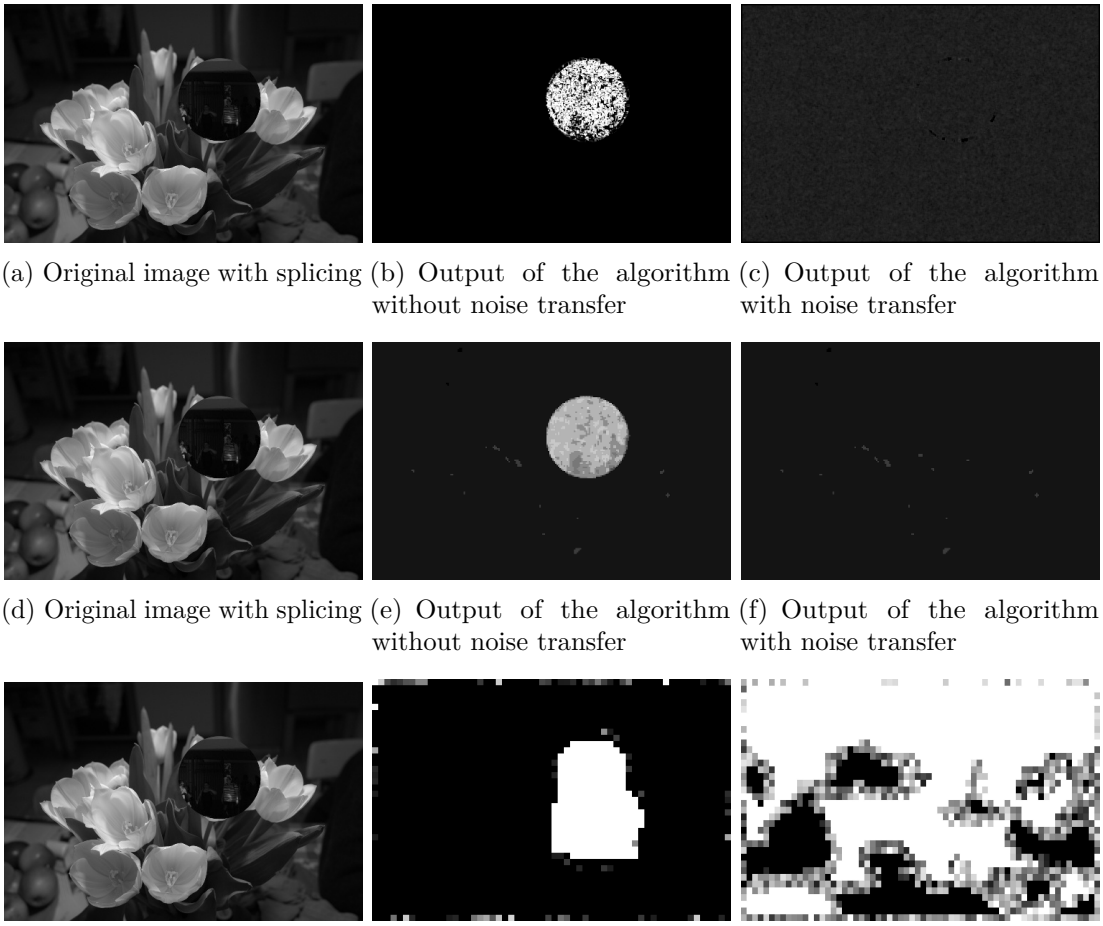
5.3.2 Test protocol

The first step of our experiments were to get an estimation of the standard deviation of the noise on the images, then use it to denoise them. The noise estimation was made using Colom and Buades work [75], and denoising was performed using the Lebrun [63] implementation of BM3D.

We tested the efficiency of our approach against three different splicing detection algorithm based on detecting noise inconsistencies: the work presented in chap-

ter 4.3, [42], and [41]. Those three algorithms use very different methods to detect splicing, which allows us to test the robustness of our approach against dissimilar detection methods. Our protocol was the same in the three cases: we splice two images without altering the noise on any of them. Then, we analyse the spliced image with the detection algorithm. Finally, we repeat this procedure, but we use our noise transfer approach before the splicing. This allows us to highlight the impact of our method.

5.3.3 Results



(a) Original image with splicing (b) Output of the algorithm without noise transfer (c) Output of the algorithm with noise transfer

(d) Original image with splicing (e) Output of the algorithm without noise transfer (f) Output of the algorithm with noise transfer

(g) Original image with splicing (h) Output of the algorithm without noise transfer (i) Output of the algorithm with noise transfer

Figure 5.6: The results on this set of images are similar to the one in Fig. 5.5.

On both Fig. 5.5 and Fig. 5.6, we can see that after being subjected to our

method, the spliced elements can not be found in the images, and this against the three detection approach we used. Indeed, in the case of Pan et al. and Mahdian et al., the spliced element noise is level with the rest of the image, and thus indistinguishable. In our case, the algorithm does seem to find that part of the image have been altered, but not in the correct position at all.

5.3.4 Consequences and discussion

As discussed in section 5.2.1, our method presents several advantages over other more naive methods. The most common naive approach is to denoise the image whose noise we want to change, and renoise it with an artificial Gaussian white noise. However, this presents several issues: first, not all images have a noise distribution that follows a Gaussian distribution. Some noises follow a Poisson distribution, or a Poisson-Gauss one. In this last case in particular, getting the parameters of the noise to reproduce it is considerably harder than with a simple Gaussian noise. Second, this method needs a denoising step, and add the noise on the denoised image. As there is no perfect denoising algorithm, the denoised image will necessarily have denoising artefacts which can take different forms, the most common being a slight blur or areas around edges that are not perfectly denoised. To our knowledge, no method exists to detect this second case, but some algorithms can already detect blurring in an image, even if it is renoised, as the one presented in [67] and [76].

In comparison, our method only uses denoising in order to compute the density histograms. All of the image modifications are made directly on the noised image. Furthermore, the fact that we select the pixel to change randomly among those eligible guarantees that there will be no observable blur or added noise around the edges.

In addition, our approach can be combined with other anti-forensics methods to hide the possible other effects of a splicing, such as the alteration of the CFA pattern.

5.4 Application: CG camouflage

5.4.1 State of the art

According to a study by Fan et al. [77], image noise is not an element that helps human to distinguish a CG from an natural image. However, several forensics methods have been developed to make this distinction.

The first relevant method to distinguish CG from natural images was introduced by Farid and Lyu [78]. They extract some features from a separable quadrature mirror filters applied on the image, and use a learning framework to classify CG from natural image. From then, nearly all the methods follow the same approach, proposing more relevant features vectors.

Among these contributions, Chen et al. [79] explored the feature extraction on HSV color space whereas concurrence only deals with each RGB channels independently.

Wu et al. [80] simplify the usual wavelet-based features by just using histograms of some measures on these wavelet decomposition. This method is one of the most accurate in the state of the art and is also the easiest to implement.

For more details, the reader can refer to Ng and Chan [81] as well as Tokuda et al. [82] who present some interesting performances evaluation tables comparing a large set of state of the art methods.

5.4.2 Consequences and discussion

In contrast to adding noise to natural images, which requires pre-emptive denoising, adding noise to CG images can be done without a denoising step, depending on the way the image was generated (some ray-tracer methods, like bidirectional path tracing, induce noise in the final image). As a consequence, there is no risk of denoising artefacts. However, this also guarantees that the resulting image will have a perfect additive white Gaussian noise, which is not found in natural images, except when very strong compression is involved. As seen in section 5.4.1, forensics methods able to differentiate CG and natural images are based on first order wavelet analysis (or similar alternatives) that is highly related to image noise.

These detection strategies exploit the unnaturalness of this perfect noise. Since our method can transfer the noise from a natural image instead of adding a generated noise, there is, in theory, a reduced chance to be detected by such algorithms. However, it is important to note that those approaches use other features beyond the pure noise statistics, and as such the impact of a more natural noise requires more research.

5.5 Conclusion

In this chapter, we have proposed a novel way to alter the noise of an image by transferring the noise density from another image. We have also presented two possible applications for this method. The first application is to dissimulate image splicing. The method is adaptable to various type of noises, and is able to fool several state-of-the-art noise-based splicing detection algorithms. The second possible application is to make computer-generated images appear less artificial, by giving them the noise of a natural image.

There is nothing concealed that will not be disclosed,
or hidden that will not be made known.

— Luke 12:1-3

6 Conclusion

Contents

6.1 Overall contribution

In this manuscript, I have presented several contribution and improvements on the state of the art in digital image forensics. The work presented in chapter 3, by analyzing and detailing the evolution of the noise throughout the camera pipeline and several base alterations, is a base building block that allows for the rest of the methods shown (and, indeed, any noise-based forensics approach) to have solid theoretical foundations. The Poisson-Gaussian nature of the noise in raw images, in particular, gave me the starting point for the first method presented in section 4.2. This method, although a bit rough along the edges, was the first noise-based method to exploit the incredible precision that working with raw images allows.

The approach presented in section 4.3, in comparison, is a much more refined and precise way to discover falsifications. Not only does it allow for increased spatial precision in the detection, it also introduces the main contribution of this work, in

the form of the tool baptized "Contribution Histograms". We have shown that this tool can, by comparing the shape of the noise in all subimages, to separate them into two categories: genuine or spliced.

This method was once again improved upon in section 4.4, with its adaptation to JPEG images. Not only does this expand the field of application to an incredible degree, it also allowed me to refine the approach further, by changing the comparison between subimages to a comparison between a subimage and the global noise. By setting a base reference to compare local noise to, the method's efficiency was drastically increased, and got better detection results than the current state of the art.

Finally, chapter 5 introduced a counter-forensics method based on the very tool I developed. Although it may seems counter-productive, since using this approach is the tailor-made counter to my own forensics algorithms (and a few others), it gives a simple goalpost for improvement: create a new method able to beat this anti-forensic approach.

6.2 Final thoughts

There is a kind of quiet jubilation in unveiling something that is supposed to be hidden. Finding the manipulation in an image squarely falls in this category, and the pleasure of detecting those alterations, even if I made them myself, has been an agreeable reward time and time again throughout this thesis. When I first started to work in the field of digital forensics, I though that I would never be able to apprehend all of it. So many ways to alter an image, and so many methods to detect each type of alteration, each one more specialized than the precedent. After working on it for the last three years, this feeling has not disappeared at all. With the constant evolution of digital cameras and their associated software, new methods of falsifications keep popping up, and associated methods of detection do the same. It is fascinating dance that needs and deserves to be continuously studied.

However, it is extremely likely that the discipline will evolve quickly in the next

few years. The impact of machine learning is already starting to be felt, and the methods using it tend to have extremely high rates of success. Since image falsification is, by nature, a manual process, it is unlikely to be able to resist for long against the emerging potential of deep learning and neural networks. It is possible, however, that image falsification evolves by itself to a new field: indeed, video falsification is now just as common. Although it probably won't develop before the next ten or fifteen years, I do believe that virtual reality alteration will be the next step, and adapting image-based methods to this new environment will certainly prove a formidable and fascinating challenge.

Appendices

A

Résumé de la thèse

Contents

A.1 Motivation

Le premier appareil photo digital, le Megavision Tessera, a été commercialisé en 1987. Depuis, le monde de la photographie digitale a été dans un état d'évolution et d'amélioration constante: temps de batterie allongé, résolution augmentée, focus et balance des blancs automatiques, divers modes convenant à tout types de situation, etc. Tous les smartphones incluent maintenant un appareil digital. En conséquence, le monde a laissé de côté la photographie argentique pour se concentrer sur les images digitales. Cependant, cette tendance montante a été accompagnée par un côté sombre: la fasification d'images.

La falsification de photographies est bien plus ancienne que l'âge du digital: par exemple, censurer ou altérer des images étaient des pratiques courantes en URSS [2], comme on peut le voir dans la Fig. A.1. Ces pratiques, cela dit, nécessitaient une formation avancée et de l'équipement spécialisé. Avec les images digitales, les seuls prérequis sont un ordinateur et un logiciel d'altération d'image, ce dernier étant

Figure A.1: Altérations successives sur un portrait des dirigeants soviétiques, au fur et à mesure que les membres perdaient la faveur de Staline.

disponible en magasin, voire parfois gratuit. En conséquence, une photographie digitale peut être modifiée par pratiquement n'importe qui. Afin de combattre cette nouvelle forme de désinformation, le domaine de l'imagerie forensique s'est développé: un moyen de détecter, *via* différentes caractéristiques des images digitales, tout type de falsifications. Les travaux présentés ici ont pour objectif de fournir de nouveaux outils pour détecter une catégorie de fasifications appelée "splicing",

en exploitant les informations fournies par le bruit inhérent à l'image. Le splicing (également appelé insertion exogène) consiste à insérer une partie d'une image A dans une image B, comme montré dans la Fig. A.2.

Figure A.2: Contrairement à ce que cette image pourrait laisser penser, l'auteur n'est pas Batman.

A.2 Contribution

La contribution principale de cette dissertation consiste en trois nouveaux outils pour détecter le splicing dans les images digitales. Elle inclue également une analyse en profondeur du bruit dans l'image et une méthode contre-forensique pour camoufler des falsifications.

Le chapitre 3 offre une étude approfondie du bruit dans les images digitales, depuis sa source jusqu'à l'image finale. En premier lieu, l'effet des différentes étapes de traitement interne à l'appareil photo (démosaiquage, débruitage interne, balance des blancs, contraste, luminosité, compression, etc) est analysé, avec une comparaison des différentes options. En deuxième partie, on étudie l'impact de méthodes de traitement d'image "classiques" (changement de résolution, compression supplémentaire, rotation, etc) sur le bruit de l'image. Enfin, une dernière section se penche sur que des falsifications fortes ont sur l'image, ainsi que sur la robustesse de diverses méthodes de forensique contre l'ajout de bruit.

Le chapitre 4.2 propose une première approche de détection de splicing, uniquement dans les images en format raw, c'est-à-dire telles qu'enregistrées par les capteurs. Cette méthode se base sur le fait que le format raw contient du bruit de Poisson-Gauss, contrairement au bruit dans une image JPEG qui est usuellement considéré comme étant un bruit blanc gaussien. Par ailleurs, les images raw ont l'avantage de ne subir aucun traitement, et donc de fournir un bruit "pur". Basé sur ces à-priori, la méthode proposée estime le bruit sur différentes zones de l'image afin d'identifer des anomalies potentielles, en clusterisant les zones selon leurs caractéristiques de bruit.

Le chapitre 4.3 introduit la base de l'outil central développé dans cette thèse, l'histogramme de contribution à la densité de bruit, ainsi qu'une première application. Comme dans le chapitre précédent, cette application se porte sur les images raw. L'outil proposé se base sur la construction de la fonction de densité de bruit de l'image (en pratique, sa représentation sous forme d'un histogramme). Si l'on suppose qu'un élément splicé a un bruit différent du reste de l'image, alors il contribuera à des zones spécifiques de la fonction de densité de bruit. En observant quels pixels contribuent à ces zones spécifiques, on pourra isoler l'élément splicé de l'image originale.

Le chapitre 4.4 étend et raffine la méthode présentée dans le chapitre précédent, en l'appliquant au format JPEG. De façon à s'adapter aux limitations imposées à la précision du bruit par le format JPEG, on transforme notre histogramme de

contribution original, sparse par nature, en une version densifiée. La forme de la base de cet histogramme est alors utilisée pour séparer nos fragments d'image en deux catégories.

Pour finir, le chapitre 5 propose une application de notre outil au domaine de la contre-forensique, c'est-à-dire le champs des méthodes utilisées pour camoufler l'altération d'une image. En connaissant les propriétés du bruit d'une image, on peut les reproduire sur l'élément splicé de façon à rendre son bruit virtuellement identique à celui du reste de l'image originale. Une application alternative est proposée pour rendre plus réaliste des images synthétiques.

B

List of publications

Thibaut Julliand, Vincent Nozick, Hugues Talbot. Automated Image Splicing Detection from Noise Estimation in Raw Images. 6th International Conference on Imaging for Crime Prevention and Detection, Jul 2015. IET Conference Proceedings, pp.13-18.

Thibaut Julliand, Vincent Nozick, Hugues Talbot. Image Noise and Digital Image Forensics. IWDW 2015, Tokyo, Japan. Digital-Forensics and Watermarking: 14th International Workshop, pp.3 - 17.

Thibault Julliand, Vincent Nozick, Hugues Talbot. Automatic Image Splicing Detection Based on Noise Density Analysis in Raw Images. ACIVS-16, Oct 2016. Springer, LNCS, International Conference of Advanced Concepts for Intelligent Vision Systems, pp.126 - 134, 2016.

Thibault Julliand, Vincent Nozick, Hugues Talbot. Countering Noise-based Splicing Detection Using Noise Density Transfer. Journal of Digital Forensics, Security and Law, the Association of Digital Forensics, Security and Law (ADFSL), 2016, 11 (2), pp.111-122.

Thibault Julliand, Vincent Nozick, Isao Echizen, Hugues Talbot. Using The Noise Density Down Projection To Expose Splicing In JPEG Images. (To be published)

References

[1] Arthur Brisbane. “Speakers give sound advice”. In: *Syracuse Post Standard* 18 (1911).

[2] David King. *The commissar vanishes: The falsification of photographs and art in Stalin's Russia*. Metropolitan Books New York, NY, 1997.

[3] Michael L Richardson, Mark S Frank, and Eric J Stern. “Digital image manipulation: what constitutes acceptable alteration of a radiologic image?” In: *AJR. American journal of roentgenology* 164.1 (1995), pp. 228–229.

[4] Mike Nizza and Patrick J Lyons. “In an iranian image, a missile too many”. In: *The Lede, The New York Times News Blog* (2008).

[5] Hailing Huang, Weiqiang Guo, and Yu Zhang. “Detection of copy-move forgery in digital images using SIFT algorithm”. In: *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*. Vol. 2. IEEE. 2008, pp. 272–276.

[6] Xunyu Pan and Siwei Lyu. “Region duplication detection using image feature matching”. In: *IEEE Transactions on Information Forensics and Security* 5.4 (2010), pp. 857–867.

[7] Sevinc Bayram et al. “Image manipulation detection with binary similarity measures”. In: *Signal Processing Conference, 2005 13th European*. IEEE. 2005, pp. 1–4.

[8] XiaoBing Kang and ShengMin Wei. “Identifying tampered regions using singular value decomposition in digital image forensics”. In: *Computer Science and Software Engineering, 2008 International Conference on*. Vol. 3. IEEE. 2008, pp. 926–930.

[9] Seung-Jin Ryu, Min-Jeong Lee, and Heung-Kyu Lee. “Detection of Copy-Rotate-Move Forgery Using Zernike Moments.” In: *Information hiding*. Vol. 6387. Springer. 2010, pp. 51–65.

[10] Matt Carlson. “THE REALITY OF A FAKE IMAGE News norms, photojournalistic craft, and Brian Walski's fabricated photograph”. In: *Journalism Practice* 3.2 (2009), pp. 125–139.

[11] Alin C Popescu and Hany Farid. “Statistical Tools for Digital Forensics.” In: *Information Hiding*. Vol. 3200. Springer. 2004, pp. 395–407.

[12] Junfeng He et al. “Detecting doctored JPEG images via DCT coefficient analysis”. In: *European conference on computer vision*. Springer. 2006, pp. 423–435.

[13] Zhouchen Lin et al. “Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis”. In: *Pattern Recognition* 42.11 (2009), pp. 2492–2501.

[14] Alin C Popescu and Hany Farid. "Exposing digital forgeries in color filter array interpolated images". In: *IEEE Transactions on Signal Processing* 53.10 (2005), pp. 3948–3959.

[15] Andrew C Gallagher and Tsuhan Chen. "Image authentication by detecting traces of demosaicing". In: *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE. 2008, pp. 1–8.

[16] Pasquale Ferrara et al. "Image forgery localization via fine-grained analysis of CFA artifacts". In: *IEEE Transactions on Information Forensics and Security* 7.5 (2012), pp. 1566–1577.

[17] Matthias Kirchner and Rainer Böhme. "Synthesis of color filter array pattern in digital images." In: *Media Forensics and Security* 7254 (2009), p. 72540.

[18] Sevinç Bayram et al. "Image manipulation detection". In: *Journal of Electronic Imaging* 15.4 (2006), pp. 041102–041102.

[19] Wen Chen, Yun Q Shi, and Wei Su. "Image splicing detection using 2-D phase congruency and statistical moments of characteristic function." In: *Security, Steganography, and Watermarking of Multimedia Contents*. 2007, 65050R.

[20] Jong Goo Han et al. "Efficient Markov feature extraction method for image splicing detection using maximization and threshold expansion". In: *Journal of Electronic Imaging* 25.2 (2016), pp. 023031–023031.

[21] Micah K Johnson and Hany Farid. "Exposing digital forgeries through specular highlights on the eye". In: *International Workshop on Information Hiding*. Springer. 2007, pp. 311–325.

[22] Micah K Johnson and Hany Farid. "Exposing digital forgeries through chromatic aberration". In: *Proceedings of the 8th workshop on Multimedia and security*. ACM. 2006, pp. 48–55.

[23] H Rand Chennamma and Lalitha Rangarajan. "Image splicing detection using inherent lens radial distortion". In: *arXiv preprint arXiv:1105.4712* (2011).

[24] Hany Farid. "Exposing digital forgeries from JPEG ghosts". In: *IEEE transactions on information forensics and security* 4.1 (2009), pp. 154–160.

[25] Matthew Stamm and KJ Ray Liu. "Blind forensics of contrast enhancement in digital images". In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE. 2008, pp. 3112–3115.

[26] J. Medkeff. "Using Image Calibration To Reduce Digital Noise In Images". In: (2004). URL: http://web.archive.org/web/20161021070914/http://photo.net/learn/dark_noise/.

[27] Roberto Costantini and Sabine Süsstrunk. "Virtual sensor design". In: *Proc. IST/SPIE Electronic Imaging 2004: Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications V*. Vol. 5301. LCAV-CONF-2004-007. 2004, pp. 408–419.

[28] Saeed V Vaseghi. *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.

[29] Hilda Faraji and W James MacLean. "CCD noise removal in digital images". In: *IEEE Transactions on image processing* 15.9 (2006), pp. 2676–2685.

[30] Anna Jezierska et al. “An EM approach for Poisson-Gaussian noise modeling”. In: *Signal Processing Conference, 2011 19th European*. IEEE. 2011, pp. 2244–2248.

[31] K Irie et al. “A model for measurement of noise in CCD digital-video cameras”. In: *Measurement Science and Technology* 19.4 (2008), p. 045207.

[32] Alessandro Foi et al. “Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data”. In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754.

[33] Miguel Colom and Antoni Buades. “Analysis and extension of the pca method, estimating a noise curve from a single image”. In: *Image Processing On Line* 6 (2016), pp. 365–390.

[34] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. “Non-local means denoising”. In: *Image Processing On Line* 1 (2011), pp. 208–212.

[35] Anna Jezierska et al. “A primal-dual proximal splitting approach for restoring data corrupted with Poisson-Gaussian noise”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1085–1088.

[36] Kostadin Dabov et al. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[37] Rastislav Lukac. *Single-sensor imaging: methods and applications for digital cameras*. CRC Press, 2008.

[38] LibRaw 0.17. *Image Decoder Library*. 2015. URL: www.libraw.org.

[39] Dmitriy Paliy et al. “Spatially adaptive color filter array interpolation for noiseless and noisy data”. In: *International Journal of Imaging Systems and Technology* 17.3 (2007), pp. 105–122.

[40] Vincent Nozick. “Camera array image rectification and calibration for stereoscopic and autostereoscopic displays”. In: *annals of telecommunications-annales des télécommunications* 68.11-12 (2013), pp. 581–596.

[41] Babak Mahdian and Stanislav Saic. “Using noise inconsistencies for blind image forensics”. In: *Image and Vision Computing* 27.10 (2009), pp. 1497–1503.

[42] Xunyu Pan, Xing Zhang, and Siwei Lyu. “Exposing image splicing with inconsistent local noise variances”. In: *Computational Photography (ICCP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1–10.

[43] Mo Chen et al. “Determining image origin and integrity using sensor noise”. In: *Information Forensics and Security, IEEE Transactions on* 3.1 (2008), pp. 74–90.

[44] Jessica Fridrich. “Digital Image Forensics Using Sensor Noise”. In: *Signal Processing Magazine* 26.2 (2009), pp. 26–37.

[45] Kurt Rosenfeld, Husrev T Sencar, and Nasir Memon. “A study of the robustness of PRNU-based camera identification.” In: *Media Forensics and Security*. Vol. 7254. 2009.

[46] Ashref Lawgaly, Fouad Khelifi, and Ahmed Bouridane. “Weighted averaging-based sensor pattern noise estimation for source camera identification”. In: *IEEE International Conference on Image Processing (ICIP 2014)*. 2014, pp. 5357–5361. URL: http://nrl.northumbria.ac.uk/17122/.

[47] S. Dehnie, T. Sencar, and N. Memon. “Digital image forensics for identifying computer generated and digital camera images”. In: *Image Processing, 2006 IEEE International Conference on*. IEEE. 2006, pp. 2313–2316.

[48] Jong-Uk Hou, Han-Ul Jang, and Heung-Kyu Lee. “Hue modification estimation using sensor pattern noise”. In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 5287–5291.

[49] Alin C Popescu and Hany Farid. “Statistical tools for digital forensics”. In: *Information Hiding*. Springer. 2005, pp. 128–147.

[50] Matthew Christopher Stamm et al. “Anti-forensics of JPEG compression”. In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE. 2010, pp. 1694–1697.

[51] Giuseppe Valenzise et al. “Countering JPEG anti-forensics”. In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE. 2011, pp. 1949–1952.

[52] Wei Fan et al. “A variational approach to JPEG anti-forensics”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3058–3062.

[53] Isaac Newton. “Letter from Sir Isaac Newton to Robert Hooke”. In: *Hist. Soc. Pennsylvania* 3 (), p. 1676.

[54] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. “Detecting digital image forgeries using sensor pattern noise”. In: *Proceedings of the SPIE*. Vol. 6072. volume. 2006, p. 15.

[55] Giovanni Chierchia et al. “A Bayesian-MRF approach for PRNU-based image forgery detection”. In: *IEEE Transactions on Information Forensics and Security* 9.4 (2014), pp. 554–567.

[56] Sean C Kelly, Robert M Guidash, and Bruce H Pillman. *Fixed pattern noise removal in CMOS imagers across various operational conditions*. US Patent 7,092,017. 2006.

[57] Henry Gordon Dietz. *Fisheye Digital Imaging For Under Twenty Dollars*. Tech. rep. University of Kentucky, 2006. URL: http://aggregate.org/DIT/PEEPFISH/.

[58] Wen Chen, Yun Q Shi, and Wei Su. “Image splicing detection using 2-D phase congruency and statistical moments of characteristic function.” In: *Security, Steganography, and Watermarking of Multimedia Contents*. 2007, 65050R.

[59] Xin He et al. “A Novel Robust Image Forensics Algorithm Based on L1-Norm Estimation”. In: *International Workshop on Digital Watermarking*. Springer. 2016, pp. 145–158.

[60] Zhongwei He et al. “Digital image splicing detection based on Markov features in DCT and DWT domain”. In: *Pattern Recognition* 45.12 (2012), pp. 4292–4299.

[61] Tian-Tsong Ng, Jessie Hsu, and Shih-Fu Chang. *Columbia image splicing detection evaluation dataset*. 2009.

[62] Graham D Finlayson, Bernt Schiele, and James L Crowley. “Comprehensive colour image normalization”. In: *European conference on computer vision*. Springer. 1998, pp. 475–490.

[63] Marc Lebrun. "An analysis and implementation of the BM3D image denoising method". In: *Image Processing On Line* 2 (2012), pp. 175–213.

[64] Thomas Gloe and Rainer Böhme. "The Dresden image database for benchmarking digital image forensics". In: *Journal of Digital Forensic Practice* 3.2-4 (2010), pp. 150–159.

[65] Hui Zeng et al. "Image splicing localization using PCA-based noise level estimation". In: *Multimedia Tools and Applications* (2016), pp. 1–17.

[66] Brian W Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405.2 (1975), pp. 442–451.

[67] Linna Zhou et al. "Blur detection of digital forgery using mathematical morphology". In: *Agent and Multi-Agent Systems: Technologies and Applications* (2007), pp. 990–998.

[68] Gaspard Monge. "Mémoire sur la théorie des déblais et des remblais". In: *Histoire de l'Académie Royale des Sciences de Paris* (1781).

[69] SS Vallender. "Calculation of the Wasserstein distance between probability distributions on the line". In: *Theory of Probability & Its Applications* 18.4 (1974), pp. 784–786.

[70] Shmuel Peleg, Michael Werman, and Hillel Rom. "A unified approach to the change of resolution: Space and gray-level". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 739–742.

[71] Jia Xu et al. "Efficient similarity join based on Earth mover's Distance using Mapreduce". In: *IEEE Transactions on Knowledge and Data Engineering* 27.8 (2015), pp. 2148–2162.

[72] Jin Huang et al. "Heads-Join: Efficient Earth Mover's Distance Similarity Joins on Hadoop". In: *IEEE Transactions on Parallel and Distributed Systems* 27.6 (2016), pp. 1660–1673.

[73] Matthias Kirchner and Rainer Böhme. "Tamper hiding: Defeating image forensics". In: *Information Hiding*. Springer. 2007, pp. 326–341.

[74] Matthew C Stamm and KJ Ray Liu. "Anti-forensics of digital image compression". In: *IEEE Transactions on Information Forensics and Security* 6.3 (2011), pp. 1050–1065.

[75] Miguel Colom and Antoni Buades. "Analysis and extension of the pca method, estimating a noise curve from a single image". In: *Image Processing On Line* 6 (2016), pp. 365–390.

[76] Gang Cao, Yao Zhao, and Rongrong Ni. "Edge-based blur metric for tamper detection". In: *Journal of Information Hiding and Multimedia Signal Processing* 1.1 (2010), pp. 20–27.

[77] Shaojing Fan et al. "Real or Fake?: human judgments about photographs and computer-generated images of faces". In: *SIGGRAPH Asia 2012 Technical Briefs*. ACM. 2012, p. 17.

[78] Hany Farid and Siwei Lyu. "Higher-order wavelet statistics and their application to digital forensics". In: *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*. Vol. 8. IEEE. 2003, pp. 94–94.

[79] Wen Chen, Yun Q Shi, and Guorong Xuan. "Identifying computer graphics using HSV color model and statistical moments of characteristic functions". In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1123–1126.

[80] Ruoyu Wu, Xiaolong Li, and Bin Yang. "Identifying computer generated graphics via histogram features". In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE. 2011, pp. 1933–1936.

[81] Tian-Tsong Ng and Shih-Fu Chang. "Discrimination of computer synthesized or recaptured images from real images". In: *Digital image forensics*. Springer, 2013, pp. 275–309.

[82] Eric Tokuda, Helio Pedrini, and Anderson Rocha. "Computer generated images vs. digital photographs: A synergetic feature and classifier combination approach". In: *Journal of Visual Communication and Image Representation* 24.8 (2013), pp. 1276–1292.