ISSN 1688-2784

Universidad de la República
Facultad de Ingeniería

The avatars of noise in digital images and their use in image forensics

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Marina Gardella

en cumplimiento parcial de los requerimientos
para la obtención del título de
Doctor en Ingeniería Eléctrica.

Director de Tesis

Dr. Jean-Michel Morel	Centre Borelli, ENS Paris-Saclay
Dr. Miguel Colom	Centre Borelli, ENS Paris-Saclay
Dr. Pablo Musé	IIE, Facultad de Ingeniería, UdelaR

Tribunal

Agnès Desolneux	CNRS, ENS Paris-Saclay
Symeon Papadopoulos	Centre for Research and Technology Hellas
Patrick Bas (revisor)	CNRS, Université de Lille
William Puech (revisor)	CNRS, Université de Montpellier
Florent Retraint (Revisor)	Université de Technologie de Troyes

Director Académico

Pablo Musé	IIE, Facultad de Ingeniería, UdelaR

Montevideo
Monday 4th March, 2024

The avatars of noise in digital images and their use in image forensics, Marina Gardella.

ISSN 1688-2784

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 215 páginas.
Compilada el Monday 4th March, 2024.
`http://iie.fing.edu.uy/`

Para mis hermanos

Esta página ha sido intencionalmente dejada en blanco.

Acknowledgements

Quisiera comenzar por agradecer a mis supervisores Miguel Colom, Jean-Michel Morel y Pablo Musé. Gracias Miguel por toda la paciencia que tuviste conmigo cuando recién había llegado a Francia. Gracias Jean-Michel por tantos valiosos intercambios, tanto a nivel académico como humano. Finalmente, Gracias Pablo, nada de esto hubiese sido posible sin tu confianza y tu apoyo.

Asimismo, quisiera agradecer profundamente a los miembros del jurado. Gracias Agnès Desolneux por haberme guiado durante todos estos años. Sin tu ayuda, nunca hubiese superado este laberinto. Gracias a Patrick Bas, Symeon Papadopoulos, William Puech y Florent Retraint por su precioso feedback sobre my trabajo. Haber tenido la oportunidad de presentarles mi trabajo a ustedes fue un gran honor. Muchas gracias por los intercambios, los consejos y los comentarios.

Durante este proceso, tuve la oportunidad de visitar la Universidad de la República (Uruguay) y la Universidad de Duke (Estados Unidos). Quisiera agradecer enfáticamente a ambas instituciones por invitarme así como a las personas con las que tuve la posibilidad de intercambiar durante éstas estadías: Matías Di Martino, Guillermo Sapiro, Gregory Randall, Matías Tailanián, Pablo Musé y Marcelo Fiori.

Agradezco también a todos los miembros del Centre Borelli. Los intercambios tenidos a lo largo de los años han enriquecido profundamente mi formación y mi calidad humana. En particular, me gustaría agradecer a Jean-Michel Morel y Gabriele Facciolo por la confianza, a Rafael Grompone, con quien tengo el placer de compartir mis tareas de enseãnza y al equipo de detección de falsificaciones (Quentin Bammey, Yanhao Li y Tina Nikoukhah) por la oportunidad de haber trabajado juntos. Quisiera agradecer también a todos aquellos que han hecho este proceso más llevadero e incluso un placer. Agradezco particularmente a Franco Marchesoni, Roger Marí, Tina Nikoukhah y Antoine Tadros. Asimismo, me gustaría mencionar a mis compañeros de Duke: Yanhao Li, Max Dunitz, Anis Ben Mabrouk, Seginous Mowlavi, Flavien Armangeon, Aitor Artola y Matthieu Serfaty, quienes me apoyaron durante el dificl proceso de redacción de esta tesis. Por último, un gran agradecimiento al equipo de secretarias del laboratorio. En particular, gracias a Alina por ayudarme siempre.

Más allá de la maravillosa gente que tuve la posibilidad de conocer gracias al doctorado, me gustaría también agradecer a todos mis amigos que, sin ser parte de mi ámbito de investigación, han sabido acompañarme en este camino. Quiero agradecer a mis amigos en Uruguay: Eliana, Nadja, Santiago, Gerardo, Diego y Maxi, por hacerme sentir que la ausencia no es tal. Y a mis amigos en París Franco, Chuchi, Manuel, Mavi, Inés y a todo el equipo CA Paris 14, por haber logrado hacer de París mi lugar.

Por último pero no menos importante, agradezco a mi familia por su apoyo incondicional, por haber sabido estar presentes a pesar de la distancia y por recordarme siempre dónde está mi hogar. En particular, quiero agradecer a mis padres y a mis hermanos, a quienes esta tesis está dedicada.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Las imágenes son potentes vectores de información que transmiten gran cantidad de datos y conocimientos a través de representaciones visuales. Su importancia en diversos ámbitos no puede negarse, ya que ofrecen ventajas únicas para la comunicación, la comprensión y la documentación. En una era caracterizada por la omnipresente influencia de las imágenes digitales, la ciencia forense de las imágenes representa una disciplina vital que aborda la acuciante necesidad de mantener la veracidad y fiabilidad del contenido visual digital. Las imágenes están naturalmente dotadas de una huella digital, incrustada durante el proceso de formación de la imagen. De hecho, la creación de una imagen digital, desde su adquisición en el sensor de la cámara hasta su almacenamiento final, imprime distintos artefactos que sirven como firma única. El objetivo de esta tesis es recuperar esta huella dactilar mediante el análisis del ruido. A lo largo de la cadena de procesamiento de la cámara, el ruido de Poisson inicial se transforma mediante múltiples operaciones propias de cada proceso de formación de la imagen, dando lugar a la imagen comprimida final. Como consecuencia, los residuos de ruido pueden arrojar importantes datos forenses.

Estos indicios permiten detectar falsificaciones. En efecto, aunque las manipulaciones actuales permiten alcanzar un alto grado de fidelidad visual, al mismo tiempo introducen alteraciones en la estructura intrínseca de la imagen. La mayoría de los métodos de detección de falsificaciones aprovechan estas alteraciones de la huella intrínseca para detectar las regiones manipuladas. La primera parte de esta tesis se centra en este problema. Aquí, proponemos dos métodos basados en la detección de inconsistencias locales del modelo de ruido con respecto a un modelo de fondo. En particular, el método Noisesniffer adopta un paso de validación *a contrario*, con el objetivo de controlar el número esperado de falsas detecciones. A continuación, exploramos la posibilidad de aprender los rastros forenses mediante redes convolucionales profundas. Por último, nos centramos en la evaluación de los propios métodos de detección de falsificaciones. Proponemos una metodología y un conjunto de datos para estudiar la sensibilidad de las herramientas de detección a rastros específicos, así como su capacidad para realizar la detección sin pistas semánticas en la imagen.

Las tareas forenses referidas a la cámara de origen, como la identificación del modelo de cámara de origen o la certificación del dispositivo de origen, también pueden realizarse utilizando dicha huella. De hecho, algunas de las huellas forenses incrustadas durante el proceso de adquisición de la imagen son exclusivas del modelo o del dispositivo. Aislando dichas señales, se puede obtener información sobre el dispositivo de origen. La segunda parte de esta tesis se centra en estas tareas. Aquí exploramos enfoques de aprendizaje para determinar si un par de imágenes contienen las mismas trazas forenses. Además, proponemos un nuevo enfoque estadístico para la certificación de la cámara de origen basado en trazas PRNU. Dicho enfoque se basa en dos pruebas de hipótesis basadas en correlaciones locales que no requieren el cálculo de distribuciones empíricas.

Aún así, nada impide a los falsificadores ocultar la huella digital de la imagen. Por eso dedicamos la parte final de esta tesis al análisis de diferentes ataques contraforenses. Destacar las limitaciones de los métodos forenses actuales es importante para saber cuánta confianza se puede depositar en una imagen y para fomentar la exploración de métodos de autenticación alternativos. Con este fin, analizamos un enfoque novedoso introducido recientemente en la literatura para eliminar los rastros de la cámaras. Este enfoque se basa en una innovadora función de pérdida híbrida durante el entrenamiento de la red definida como una combinación de tres funciones diferentes: la función de similitud incrustada, la función de fidelidad truncada y la función de identidad cruzada. Además, proponemos un nuevo ataque contraforense basado en modelos de difusión.

Contents

I Forgery detection based on the alterations on noise 41

Chapter 1

Introduction

This chapter provides a short description of the problem and the main contributions of this thesis. The focus is the analysis of the noise in digital images and its forensics applications. Throughout this dissertation, we present not only the problem of forgery detection but also other forensics tasks based on noise characteristics such as source camera model identification and source camera certification. Furthermore, we explore the robustness of such approaches to different counter-forensics attacks. This chapter ends by listing the contributions of this thesis through publications, online demos, projects, and transitions to society.

Parts of the first section of this chapter have been published as a book chapter [12, 183]. The considerations presented in the final section are part of the work published as *The approach to reproducible research of the Image Processing On Line (IPOL) journal* in the Informatio journal [174].

1.1 The social maze of fake images: challenges and strategies

The Internet, digital media, new means of communication, and social networks have boosted the emergence of a connected world where the idea of achieving absolute control over information seems unattainable. Images are ubiquitous and therefore have become an essential part of the news. Unfortunately, they have also become a tool of disinformation aimed at distracting the public from reality.

Manipulation of images is everywhere. Simply removing red eyes from family photos could already be considered an image manipulation, whereas it is simply aimed at making a flash image look more natural. Even amateur photographers can easily erase the electric wires hanging from their poles in a vacation panorama, correct cosmetic imperfections such as wrinkles on a face, not to mention touch-ups done on models in magazines.

Beyond these mostly benign examples, image manipulation can lead to falsified results in scientific publications, reports, or journalistic articles. Altered images might imply an altered meaning, and can thus be used as forged evidence, for instance, to defame a person or even to report UFO or paranormal activity. More frequently, falsified images are published and relayed on social media, in order to produce and spread fake news.

The proliferation of consumer software tools and their ease of use have made image manipulation very accessible to many users. Some software even goes as far as to automatically restore a natural look to an image when parts of it have been altered or deleted. Recently, deep neural networks have made it possible to forge images almost automatically. One example is the site This Person Does Not Exist[1], which randomly generates surprisingly realistic faces of people who are not real. The most surprising application is, undoubtedly, the arrival of deepfake methods, which allow, among other features, to replace a face in a video with one of another person (face swapping).

The digital image is an essential medium of communication in today's world. People need to be able to trust this method of communication. Therefore, it is essential that news agencies, government agencies, and law enforcement maintain and preserve trust in this essential technology.

1.1.1 Criminal background

These new possibilities of image manipulation have been exploited for a long time by governments, criminal organizations, and offenders. Stalinist propaganda images come to mind, as certain characters who had become undesirable were removed from official photographs, as we can see in Figure 1.1.

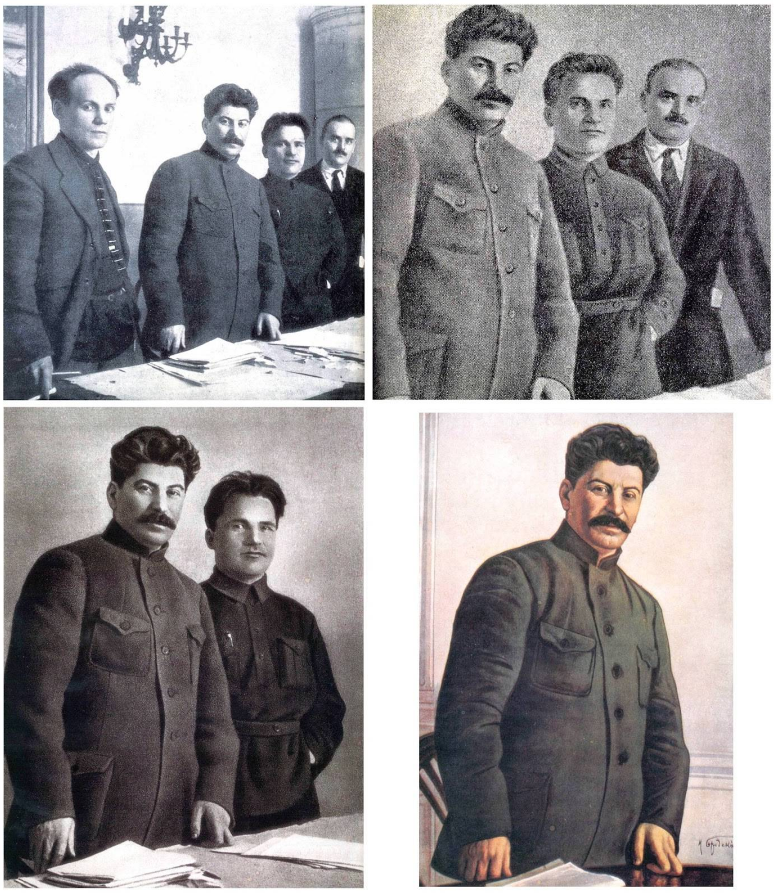


Figure 1.1: An example showing successive modifications applied to an image.

Today, image manipulation can serve the interests of criminal or terrorist organizations as part of their propaganda (false claims, false events, masking of identification elements, addition of objects). Face swapping and deepfake techniques are also a simple way to undermine the reputation and privacy of public figures by placing them in compromising photos or videos. The manipulation of images is also a means of exerting coercion, pressure, or blackmail against a

[1] www.thispersondoesnotexist.com.

third party. These new image manipulation techniques are also used by pedophiles to generate photographs that satisfy their fantasies.

Manipulated images can also be used to cause economic harm to companies through disinformation campaigns. Administrative documents can be falsified in order to obtain official papers, rental documents, or a loan from specialized organizations. Face morphing, whose objective is to obtain the photo of a visually "compatible" face from two faces, enables two users to share the same ID in order to deceive an identity check.

1.1.2 Issues for law enforcement

In the past, confessions, testimonies or photographs were enough to prove guilt. Technology was not sufficiently developed to mislead investigators. Today, these methods are no longer sufficient and law enforcement authorities need innovative scientific tools to be able to present reliable evidence in court. As technology evolves rapidly, law enforcement agencies must continuously ensure scientific monitoring in order to keep up with state-of-the-art technology, anticipate, and have the most recent tools available to detect manipulation and other forms of cheating for malicious purposes. It is essential to maintain a high level of training for the experts responsible for authenticating the images. In fact, the role of the police, and in particular of the technical and scientific police, is to highlight any falsification in order to allow perpetrators to be sentenced but also to exonerate the persons under judicial inquiry if they are innocent or if their crime cannot be proven. The role of the expert in image authentication is to detect any form of manipulation, rigging, or editing aimed at distorting reality. They must be able to answer the following questions:

- Is the image authentic?
- Does it represent the real context of the depicted scene?
- What is the history of the image and its possible manipulations?
- Where is the manipulated part?
- Is the image coming from the actual device that supposedly produced it?

In general, it is easier to conclude that an image is falsified than to say it is authentic. Detecting manipulation traces is getting harder over time, as new forgery methods are being developed. As a consequence, not finding any forgery traces does not prove the image's authenticity. The level of expertise of the forger should also be taken into account. In fact, the possible traces of manipulation will not be the same depending on whether the author is a neophyte, a seasoned photographer, or a special effects professional. The author can also use so-called anti-forensic techniques aimed at masking traces of manipulation so that they become undetectable by experts; it is up to the expert to know these techniques and their weaknesses.

1.1.3 Current methods and tools of law enforcement

As technologies evolve over time, detection tools must also adapt. Particularly during the transition from film photography to digital images, the authentication methods that were mainly based on a magnifying glass observation (visual analysis of defects, consistency of shadows and lighting, vanishing points) have been completed with structural and statistical analyses.

To this date, few commercial tools can authenticate images effectively. Most of the time experts need to design their own tools, which poses the problem of their acceptability in court. In order to compensate for this lack of objective and precise tools, the police recruit trainees, who participate in national projects (for example, the DEFALS challenge funded by French DGA[2] and the National Research Agency) or international projects (H2020 projects of the European Commission). The objective is to involve university researchers as well as industrial actors and practitioners (forensic experts). In addition, experts are developing good practices guides such as the "Best Image Authentication Practice Manual" within the framework of the ENFSI[3], in order to standardize and formalize analysis methodologies.

1.1.4 Issues for journalists

Verifying images has become a major part of the journalists' everyday job to quote and reuse eyewitness content or to debunk decontextualized and tampered pictures. Nevertheless, proving the authenticity of images found on the web remains a challenging task.

Following the rise of the so-called *fake news* wave in 2016, fact-checking has become very trendy among media organizations and NGOs. The database maintained by the US Duke University reporters' Lab lists in August 2021 a total of 349 active fact-checking organizations in the world.

Social media giants such as Facebook have partnered with fact-checkers to help them verify viral content on their platforms, including images and videos. Fact checkers need therefore to be able to prove and explain in a scientific and verifiable process, why an image is fake. The first common step is to reverse search with an engine indexing billions of images such as Google images, Yandex, Bing or Tineye. The image may be real but simply taken out of context (date, place, depicting another previous event).

Fact-checkers may also find an original image (or a supposed original image). Then, they need to match and compare this image with the one they try to verify. If no original image can be found, then the only possible method would be to detect hypothetical forgeries in the image itself, rather than the binary file.

1.1.5 Current methods and tools for journalists

Even if the research field of digital image forensics has a strong link with the fight against fake news, the developed methods usually remain in the academic environment. Indeed, most of these methods are unknown or difficult to use by the general public. Their implementation -when not provided by the authors- often requires background knowledge of image processing and coding skills. And, even when the implementation is provided, making the algorithms run still requires some computational expertise. Some academic tools try to bring these methods into the public domain (such as the demo platform of the IPOL journal) but their use by the general public is still underdeveloped.

In order to close this gap, different image verification tools have been created. These platforms are specially created for general public use, helping fact-checkers and individuals in general to integrate the forensics methods developed by academia in their daily lives. We can mention as the main image verification tools the following:

- The Image Verification Assistant [229] is a web-based application, developed within the REVEAL project, that exposes the results of seven image forensics algorithms to end users, and additionally presents the EXIF metadata (if any) of the input image.

[2] Direction Générale de l'Armement.

[3] ENFSI: European Network of Forensic Science Institutes.

- The InVID-WeVerify plug-in [204] incorporates seven state-of-the-art forensic methods to analyze still images, using the backend of the above Image Verification Assistant. Furthermore, it integrates image-reverse search engines, metadata viewers as well as a magnifier lens. This platform also performs video analysis by keyframe fragmentation.

- FotoForensics is an online platform that provides a simple interface for image analysis. The list of integrated tools includes metadata extraction and error level analysis, which consists in recompressing the image at a known uniform compression quality and analyzing the residual between these two. Despite being easy to use and free, it does not incorporate actual forensic algorithms but rather metadata readers and visualization tools.

- The Forensically online tool regroups a set of filters for digital image forensics. The main features include metadata extraction, error level analysis, noise level estimation, luminance gradient computation, and JPEG analysis. Filters are provided together with parameters the user can adjust. This platform is free and easy to use. However, despite being more complete than the previously described one, it does not incorporate most of the recent state-of-the-art forensic filters.

- Ghiro is an open-source project that provides a fully automated image forensic tool. The main features are metadata extraction, thumbnail consistency analysis, GPS localization, error level analysis, and image hash matching. Despite being open source, local installation is not straightforward for the general public. Furthermore, the last update dates from 5 years ago.

- The Assembler experiment conducted by Jigsaw and Google Research provided journalists and fact-checkers with recently developed methods to detect manipulated images. It incorporated six state-of-the-art filters, combining both AI models and classical methods. This experiment is now closed according to Jigsaw‘s website.

- The Authenticate software by Amped provides a comprehensive tool for image analysis. It includes integrity verification, context analysis, camera identification, processing analysis, and tampering detection. However, this is a professional expensive software, inaccessible to the general public.

In a survey launched at the beginning of the Envisu4 project, fact-checkers operating forensic tools answered that they mainly use the InVID-WeVerify verification plug-in forensic toolkit (96,8%) but also Forensically (28,6%), Fotoforensics (25,4%) and the (now closed) Assembler experiment from Google Jigsaw (4,8%).

It is worth mentioning that verification platforms are not the only support fact-checkers use to analyze images. Reverse image search engines such as Google, Yandex and TinEye, are also used to analyze and compare visually similar images on the web.

In the next section, we go deeper into the image formation process to reveal the traces such operations leave in the final image.

1.2 The camera processing pipeline and its traces on noise (Chapter 2)

From the raw acquisition on the camera sensor to its storage, an image undergoes a series of operations: denoising, demosaicing, white balance, gamma correction, and compression, as depicted in Figure 1.2. A description of each of these steps is provided in Section 2.2.

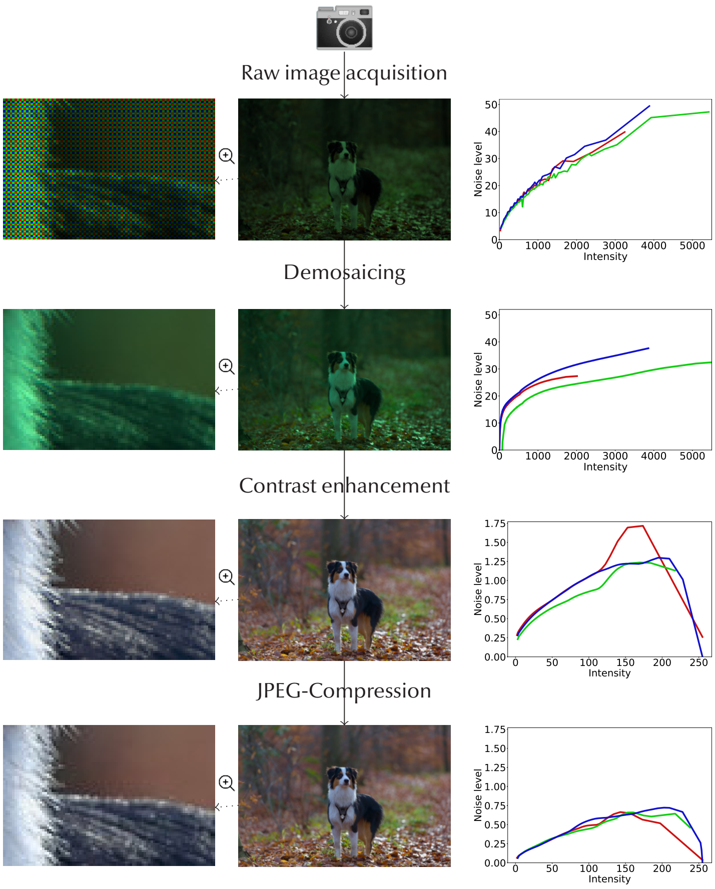


Figure 1.2: Simplified processing pipeline of an image, from its acquisition by the camera sensor to its storage as a JPEG-compressed image. The central column represents the image as it goes through each step. The left column shows the details of the image obtained as it goes throughout the camera processing pipeline. The right column plots the noise of the image as a function of intensity in all three channels (red, green blue) [41]. Because each step leaves a specific footprint on the noise pattern of the image, analyzing this noise enables us to reverse-engineer the pipeline of an image. This, in turn, enables us to detect regions of an image that were processed differently and are thus likely to be forged.

These operations leave traces on the final image, often imperceptible to the naked eye but still detectable. By analyzing those artifacts, it is possible to reconstruct the history of an image. Indeed, one can model the different operations that took place during the creation of the image, as well as their order and parameters. Being able to model the image creation pipeline of an image is relevant by itself, in particular, because it can guide the restoration of the image. More importantly, it serves as a distinctive fingerprint for the image, an intrinsic watermark. Discrepancies within the pipeline that lack coherence throughout the entire image frequently offer valuable clues of tampering.

As it can be seen in Figure 1.2, each step in the camera processing chain leaves traces in the noise model. Indeed, the raw image captured by the camera sensor follows a Poisson noise model which is, furthermore, identical for all the color channels. Once the image is demosaiced, such a model is no longer valid. Even more, the different color channels have their own different noise curve. Gamma correction gives the noise curve a bell-shape, due to the power law function this operation applies. Finally, JPEG compression flattens the bell-shaped curve. All this process is presented more in-depth in Sections 2.3 and 2.4.

Noise analysis offers a relevant insight into the image history. Indeed, if part of an image has been locally modified, or comes from a different donor image, the authentic and forged regions are thus likely to present different noise profiles. Figure 1.3 depicts this situation: the forged region presents a different noise model than that of the background image. Section 2.5 briefly reviews the use of such camera traces to develop forgery detection methods in the literature.

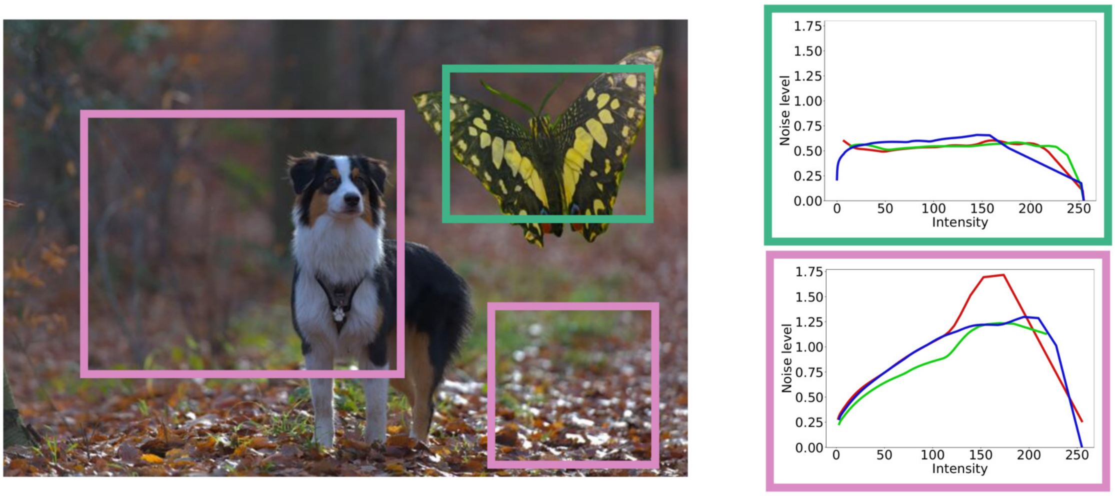


Figure 1.3: Example of a forged image (left) and local noise curves (right). The forged area comes from a different image that has its own pipeline. Noise models (right) differ between the background image (pink) and the donor one (green). The resulting tampered image presents local inconsistencies in the noise model.

From the considerations formulated in Chapter 2 and presented in this section, we developed the forgery detection method described in Chapter 3 which we shall introduce in the next section.

1.3 Forgery Detection in Digital Images by Multi-Scale Noise Estimation (Chapter 3)

As we saw in the preceding section, noise analysis offers a relevant insight into the image history. However, the final noise model present in JPEG-compressed images is complex since it is not only signal-dependent but also frequency-dependent (see Section 2.4 for more details). To deal with such complexity, in Chapter 3 we propose a multi-scale algorithm. The multi-scale approach has been shown to effectively deal with the correlations introduced by the demosaicing and JPEG-compression processes [42] and stands out as a suitable framework for noise inconsistency analysis.

The proposed approach consists of computing local noise curves and comparing them to the global noise curve, obtained from the whole image, computed using the extended version of Ponomarenko *et al.* method [41]. For non-forged images, we expect local noise curves to show similar noise level functions as the global one. However, noise estimation is highly affected by image content. Indeed, noise overestimation is expected to happen in textured regions [146]. As a consequence, local noise curves computed over textured areas may be above the global one, even if no tampering has been performed. To prevent this kind of region from being perceived as suspicious, we only consider suspicious local noise curves that are *below* the global one. Indeed, the global noise curve provides a lower bound for local noise curves since the noise estimation algorithm [41] has more samples from which to choose the adequate ones to estimate noise.

This process is done for each channel. Each color channel provides a heatmap showing the percentage of bins of each region that are below the global noise curve, as depicted in Figure 1.4. The information of the three channels is combined by taking the mean of them.Such a process is repeated through several scales in order to capture the noise inconsistencies in low and medium frequencies. Finally, the heatmaps at each scale contribute with the same weight to the final output, as depicted in Figure 1.4.

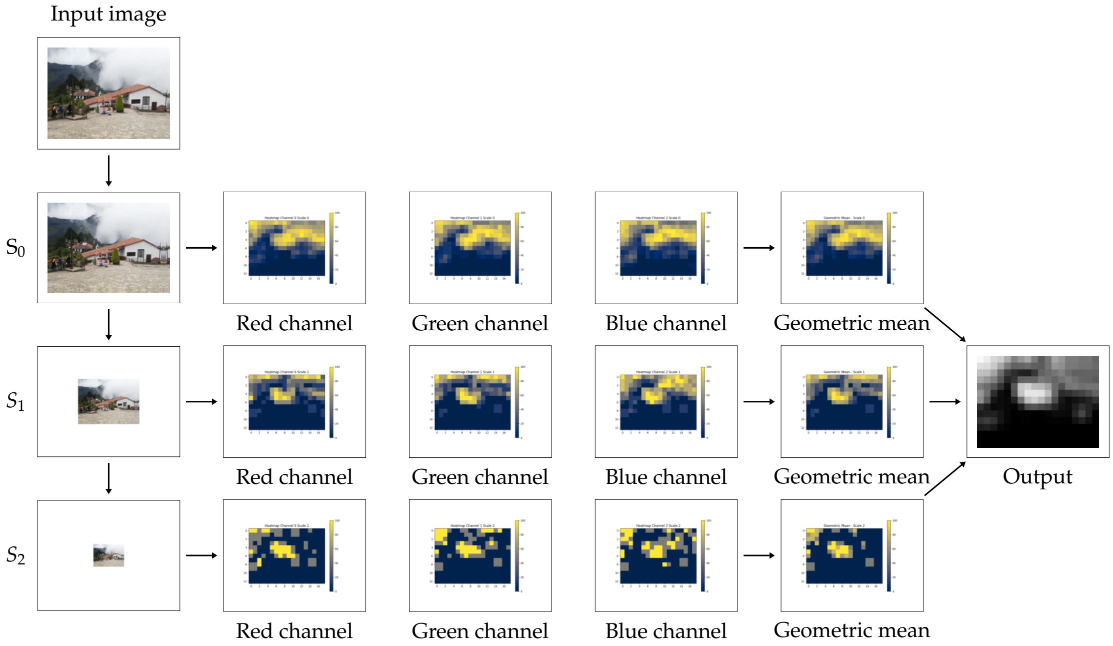


Figure 1.4: Complete pipeline of the method: successive scales are extracted from the input image. At each scale, one heatmap per color channel is computed and then combined according to their geometric mean. Finally, the obtained heatmaps at each scale are summed and normalized to produce the final output.

This method, although simple, is able to outperform state-of-the-art methods for colorization attacks. This forgery technique shows the relevance of considering intensity-dependent noise models instead of single noise levels. Indeed, when changing the color in a region of the image, noise levels are not necessarily perturbed. However, those noise levels will not be consistent with the new intensity but with the original. Estimating noise curves as the proposed method does enable detection of this kind of inconsistency which only appears when considering intensity-dependent noise models.

Besides, we evaluate the pertinence of the multi-scale approach. Using multiple scales leads to better results compared to a single one. Regarding the number of scales yielding a better performance, the use of three scales gives the best scores. In fact, given that JPEG compression is applied in 8×8 blocks without overlap, it is at the third scale that we are able to capture noise contained in lower frequencies, less affected by the JPEG quantization in the DCT coefficients.

In the next section, we shall introduce another forgery detection method based on noise analysis. Although this method is more sophisticated than the one presented here, it is based on the principles of the noise estimation method [41] presented here.

1.4 Noisesniffer: Forgery Detection by Noise Spatial Statistics (Chapter 4)

In Chapter 4 we present a more sophisticated image forgery detection method based on noise analysis than the one presented in Chapter 3. This method is based on the principles of the noise estimation method [41] used in Chapter 3. Indeed, most non-parametric noise estimation methods share the same principles: they start by selecting the homogeneous regions of the image where noise dominates over the signal, and then they estimate noise in the frequency or the spatial domain using just a small portion of the blocks in the previously selected region [132].

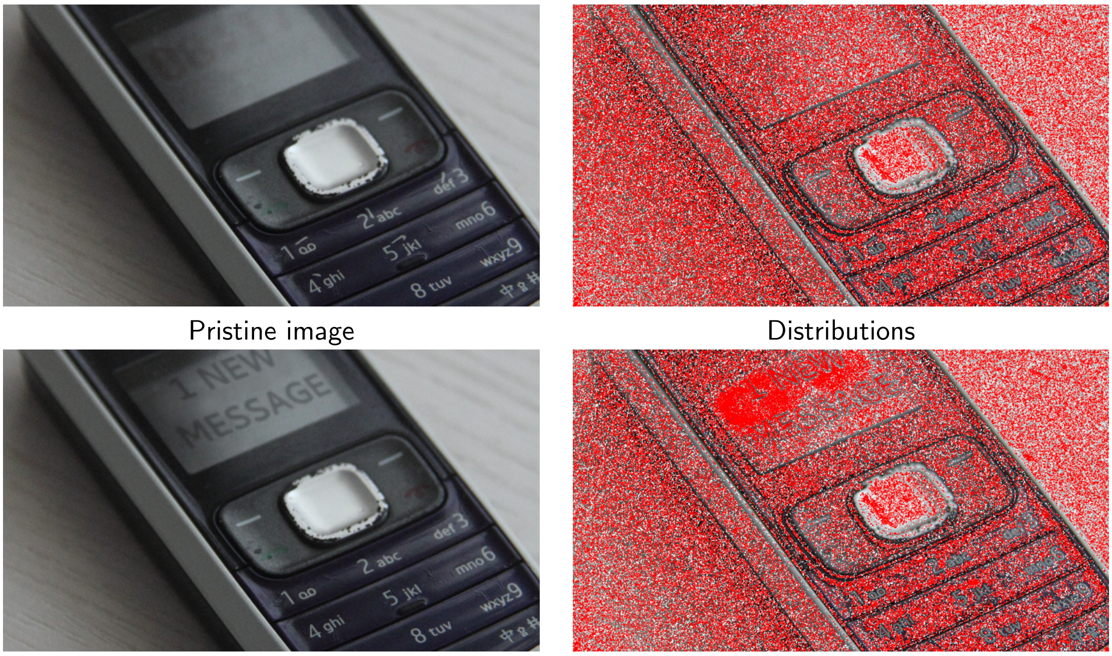
The presented method follows a similar procedure to identify the blocks within the homogeneous regions that are good candidates for noise estimation. However, instead of using these blocks to estimate the noise level, we are interested in analyzing the spatial distribution of such blocks. Indeed, if the variance in the homogeneous regions identified in the first step is only explained by noise, the small proportion of blocks that are then used to estimate noise should be a random uniform selection over the homogeneous region. However, if we observe that the blocks used for noise estimation concentrate in a particular part of the homogeneous region, we can deduce that this zone exhibits a suspicious noise deficit.

The proposed method is developed in two steps. Firstly, we compute the set of blocks corresponding to the homogeneous regions and then the subset of those blocks having the lowest standard deviation. Secondly, the spatial distribution of these two sets is compared to detect if there is a statistically significant deviation from one distribution to the other. To do so, we adopt an a contrario approach [56] and assign a number of false alarms (NFA) to each detection.

In order to select the suitable blocks for noise estimation, for each color channel and intensity, we compute their variance in low and medium frequencies and only keep a small percentile of those. The intensity variations in these blocks are likely to be explained only by noise. Then, for all these blocks we compute their variance and compare the spatial distribution of the whole set of blocks to the one of those ones having the lowest variance. As depicted in the top row of Figure 1.5, in absence of any forgery, this subset should correspond to a random uniform selection over the whole set of blocks previously selected. However, as shown in the bottom row of Figure 1.5, forgeries inducing noise deficit cause deviations from one distribution to the other.

Still, deviations from one distribution to another might happen just by chance. Thus some criterion is needed in order to spot deviations that are statistically significant from those that could happen just by chance. In Chapter 4, we propose to use an a contrario approach [56] to statistically validate these deviations. This theory is based on the Non-Accidentalness Principle, which states that we perceive a structure whenever a large deviation from randomness occurs [8]. However, computing the probability of these events might be hard. This problem is solved by the introduction of the Number of False Alarms (NFA) of an event, which is an upper bound on the expectation of occurrences of such event under the null model [56]. By establishing the desired threshold on the NFA –which corresponds to an estimate of the mean number of false detections under the background model – we can obtain automatic forgery detection masks. Figure 1.6 shows the obtained masks for the example previously depicted in Figure 1.5.

Noisesniffer is first evaluated in Section 4.4 using a subset of the Trace database, which is introduced in Chapter 6. Then, in Chapter 6 we present a more exhaustive evaluation. In addition, we offer an online executable version of the described method allowing everyone to test it on their own suspicious images https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000341.



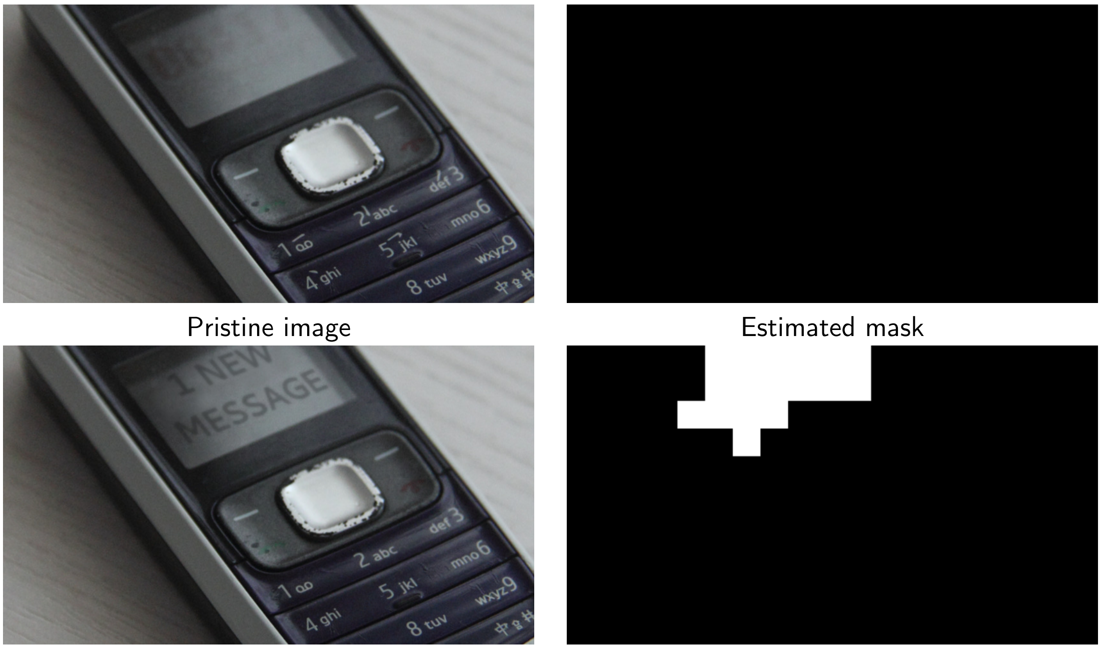
Pristine image

Distributions

Forged image

Distributions

Figure 1.5: Distributions computation results in pristine (top) and forged (bottom) images from the Korus dataset [121, 122]. In white, the 3×3 blocks having the lowest variance in low and medium frequencies. For these blocks, the intensity variations in these blocks are likely to be explained only by noise. In red, a small percentile amongst this set having the lowest variance. In the absence of any forgery, this subset should correspond to a random uniform selection over the whole set of blocks previously selected. However, forgeries inducing noise deficit cause deviations from one distribution to the other.



Pristine image

Estimated mask

Forged image

Estimated mask

Figure 1.6: Forgery detection masks obtained for the example previously shown in Figure 1.5 by detecting deviations from the background model that are statistically significant [56].

1.5 Exploring Image Forgery Detection via Forensic Similarity Graphs (Chapter 5)

The two methods presented in the previous sections aim at reconstructing the image formation process by detecting visually imperceptible traces that are left in the image at each step of the camera processing pipeline. Indeed, most classical methods rely on hand-crafted features. However, learning camera-related features has also been attempted in the literature. In Chapter 5, we explore one of such approaches: the forensic similarity network introduced by O. Mayer and M.C. Stamm and its applications to forgery detection.

To construct the forensic similarity score, Mayer and Stamm propose to design a feature extractor function f and the similarity function S such that $S(f(X_1), f(X_2))$ is as close to 1 as possible whenever the patches X_1 and X_2 have the same camera traces and as close as possible to 0 when not. To develop such functions, they propose a learning-based strategy.

The feature extractor consists of 5 convolutional blocks and 2 fully connected layers. Two such feature extractor networks, in siamese configuration with weight-sharing, are used to process in parallel the two patches, producing a feature vector for each patch. Then, a similarity network takes both feature vectors as input and computes their similarity score.

The system is trained in two phases. In the first phase, the feature extractor is trained by adding a fully connected layer with softmax activation. The feature extractor is trained as a source camera classifier, using image patches with associated labels corresponding to their source camera model. Research indicates that the deep features associated with camera model classification provide a good starting point for several forensics tasks. During the second training phase, the similarity network is trained to target a specific task. Here, the task is to determine if two image patches come from the same camera model.

By computing the forensic similarity of each of the patches within an image, the authors construct a graph representation where patches are represented as the vertices of the graph, and the edge weights are assigned in order to reflect the forensic similarity between them. Patches that have undergone the same processing operations are expected to form communities. Communities are characterized by strong connections within the members and weak connections to non-members. This way, forgery detection can be formulated as a community detection problem, and forgery localization as a community partition problem.

In order to detect communities, the image graph representation is modeled as a weighted graph $G = (V, W)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and W is a symmetric matrix satisfying $W_{ij} \geq 0$ for all $i, j = 1, \ldots, n$ and $W_{ii} = 0$ for all $i = 1, \ldots, n$. Two approaches are then proposed to detect communities in weighted graphs: spectral clustering [66] and modularity optimization [171].

Spectral clustering focuses on the study of the eigenvalues of the graph Laplacian matrix, defined as

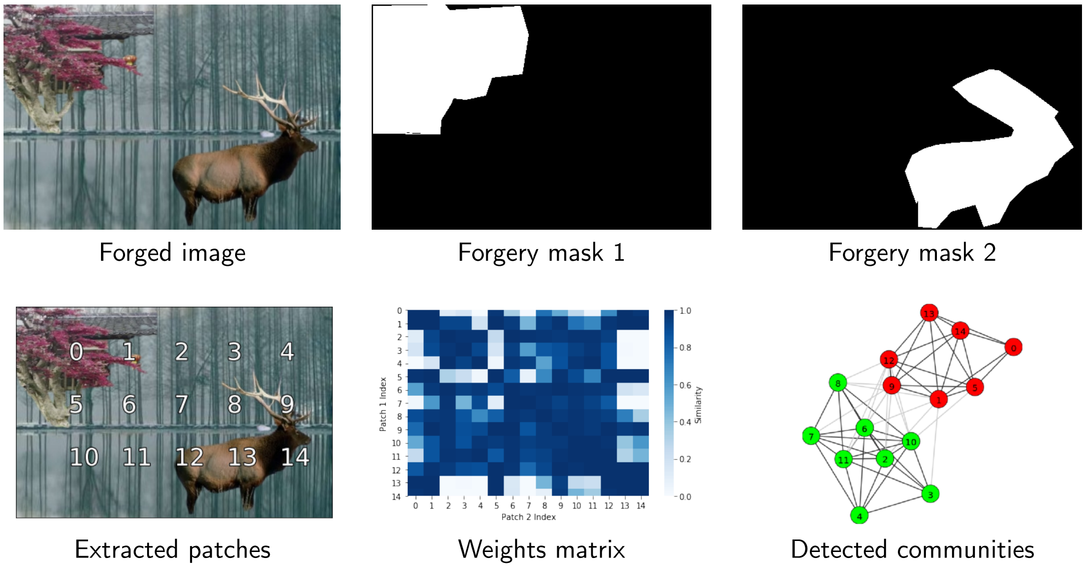
$$L = D - W \tag{1.1}$$

where D is the degree matrix defined as $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ if $i \neq j$. The multiplicity of the eigenvalue $\lambda = 0$ corresponds to the number of communities in the graph. The authors use the *eigengap heuristic* [153] to detect if more than one community exists by computing the second smallest eigenvalue λ_2, and comparing it to a pre-defined threshold. If λ_2 is smaller than τ, the image is classified as forged, and if it is larger, as non-forged.

On the other hand, modularity was introduced as a measure of the quality of a particular graph partition [171]. This is done by comparing the observed edge density within a community to the expected edge density given by the background model. Modularity can be expressed as

$$Q = \frac{1}{2m} \sum_{i,j} (W_{ij} - E(W_{ij}))\, \delta(c_i, c_j), \tag{1.2}$$

where the sum is taken over all the pair of vertices, m is the weighted total number of edges $m = \frac{\sum_{i,j} W_{ij}}{2}$, W is the weights matrix, $E(W_{ij})$ is the expected weight for an edge connecting vertices i and j given by the background model, δ is the Kronecker δ-function and c_i is the community to which vertex i belongs. Modularity optimization aims at finding the community partition for which the modularity is maximized. If the optimal value for the modularity is close to 0, then there is no evidence of community structure [171]. Therefore, to detect image tampering the authors compare Q^{opt} to a pre-defined threshold.



Forged image

Forgery mask 1

Forgery mask 2

Extracted patches

Weights matrix

Detected communities

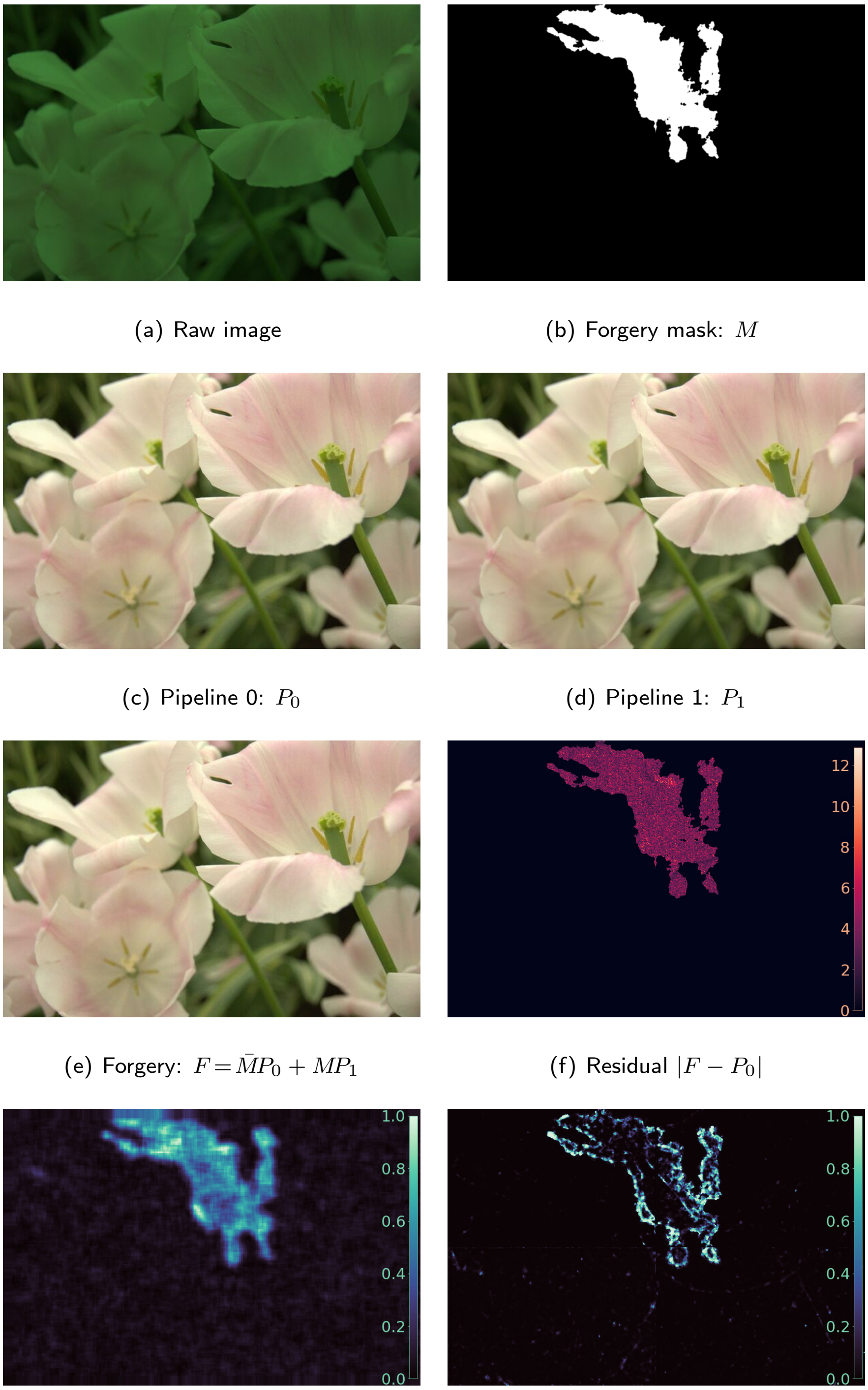
Figure 1.7: Examples of weight matrices and graph partitions on a forged image from the MISD Dataset [109], having two spliced regions. The community detection algorithm used for these examples is the modularity optimization, with patches of size 128×128 and 50% of overlap. The edge weight threshold was set to 0.9. We observe that the community partition found by the algorithm (in red) points to the forgery masks.

Figure 1.7 shows an example of the previously described detection method. In the top row, the input image and its corresponding splicing masks. In the second row, we observe the extracted patches, the weight matrix associated with the graph representation of the image and, finally, the communities partition found using modularity optimization.

In Chapter 6 we present an exhaustive quantitative evaluation of the described method. In addition, we offer an online executable demo allowing everyone to test it on their own suspicious images: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=432.

1.6 Non-Semantic Evaluation of Image Forensics Tools (Chapter 6)

In the previous sections, we have focused on image forgery detection methods. In order to evaluate their performance, a key decision is the evaluation framework. In this section, we shall describe a new evaluation approach.



(a) Raw image

(b) Forgery mask: M

(c) Pipeline 0: P_0

(d) Pipeline 1: P_1

(e) Forgery: $F = \bar{M}P_0 + MP_1$

(f) Residual $|F - P_0|$

(g) Noiseprint result

(h) Mantranet result

Figure 1.8: Different image formation pipelines are applied to the same RAW image to obtain two images, that are combined to obtain a forged image. The authentic and forged regions present different camera pipeline traces but are otherwise perfectly coherent. The last row shows the result of two forensic tools on this image: Noiseprint [48] and Mantranet [221].

Image forensics algorithms are mainly evaluated by their performance in benchmark challenges. This practice has several limitations: in many cases, the same database is split into training and evaluation data. As a consequence, algorithms are trained and evaluated on images that have gone through similar image processing pipelines, forgery algorithms, and anti-forensic tools. Hence, there is no guarantee that such learning-based methods will work in the wild, where those parameters vary much more. Regardless of the variety of the training set, the question arises of whether the forgeries are being detected by trained detectors for semantic reasons, or because of local inconsistencies in the image.

Indeed, while semantic analysis of an image can provide hints, the rigorous proof of a forgery should not be based on semantic arguments only. The situation is similar to the dilemma arising from the observations of Galileo, which contradicted the accepted knowledge of his time. In the words of Bertolt Brecht [24]:

> GALILEO: How would it be if your Highness were now to observe these impossible as well as unnecessary stars through this telescope?
> THE MATHEMATICIAN: One might be tempted to reply that your telescope, showing something that cannot exist, may not be a very reliable telescope, eh?

The telescope could have been unreliable, indeed, and a scientific inquiry on the instrument could have been justified. However, concluding, as the Mathematician does, that the telescope was unreliable *just* based on the contents of the observations is not prudent. Similarly, the proof of a forgery must be based on image traces, not on semantic arguments, because the semantics of an image are usually the purpose and not the means of a forgery.

With these considerations in mind, in Chapter 6 we propose a methodology and a database to evaluate image forensic tools on images where authentic and forged regions only differ in the traces left behind by the image processing pipeline. Using this methodology, we create the *Trace database* by adding various forgery traces to raw images from the Raise [52] dataset, as shown in Figure 1.8. This procedure avoids the difficulties of producing convincing and unbiased semantic forgeries, which often require manual work. We create several datasets, each of which corresponds to a specific pipeline inconsistency, such as a different noise level or compression pattern. This gives us insight into the sensitivity of forensic tools to specific traces and thus highlights the complementarity of different methods.

1.7 Analysis of the Forensic Similarity Approach for Source Camera Model Comparison (Chapter 7)

As pointed out in Chapter 2, the traces left by the image processing pipeline can be used for several forensics tasks rather than image forgery detection and localization. In this chapter, we explore another use of such traces related to source camera forensics. Providing information about the camera with which an image was acquired can be crucial for different forensic applications. Indeed, it can provide clues to track pornographic content, check for copyright infringement, and verify the consistency of a database.

Classic methods tackle this problem by searching for device traces. These traces include sensor pattern noise [151], lens distortions [36], demosaicing artefacts, white balance traces [55] and compression. Some of these features, such as the PRNU[4] pattern [151] or radial distortion [36], are device-specific and can lead to accurate device identification. In particular, the use of PRNU traces for source camera certification is explored in Chapter 8. Other features

[4] Photo Response Non-Uniformity

are shared by different devices from the same model or brand, and can therefore provide information about the device model rather than identifying a particular source camera [207].

In this chapter, we show that the forensic similarity approach presented in Chapter 5 can also be used for source camera model comparison. Indeed, such an approach aims to determine whether two image patches share the same forensic traces or not. Images acquired with devices from the same model are expected to exhibit the same similar forensic traces, while devices from different models are expected to produce different traces. In order to make the chapter self-contained, the forensic similarity approach is presented again in Section 7.2.

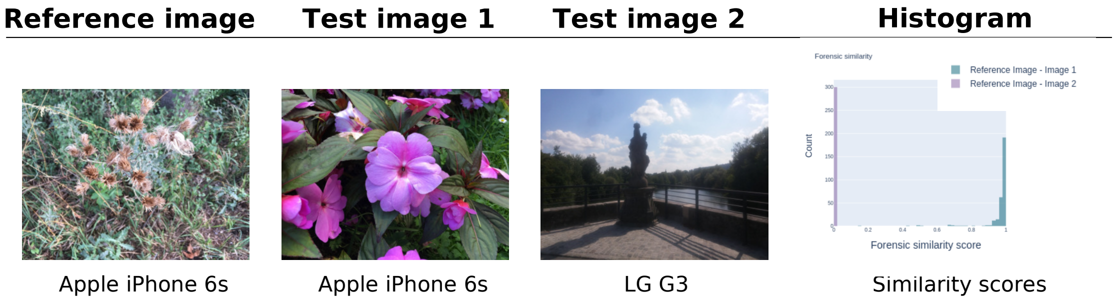


Figure 1.9: Results of the forensic similarity approach applied to source camera comparison. The first figure presents the reference image, the second and third the test images and, finally, the fourth figure shows the histogram of the forensic similarity scores obtained in the patch-to-patch comparisons.

The goal of the method is to determine if a pair of images has been captured by the same camera model or not. The method takes as input three images: a reference image, and two test images that will be compared to the reference one. To perform an image-wise comparison built upon the patch comparison provided by the forensic similarity approach, we set a number of randomly chosen patch-to-patch comparisons to be considered. The output is a histogram showing the forensic scores obtained for each patch-to-patch comparison, for both image comparisons. Figure 1.9 shows an example of the forensic similarity approach applied to source camera model comparison. We offer an online executable version of the described method allowing everyone to test it on their own suspicious images https://ipolcore.ipol.im/demo/clientApp/demo.html?id=424.

1.8 Photo-Response Non-Uniformity (Chapter 8)

Following the same approach introduced in the previous section, Chapter 8 also focuses on source camera forensics based on the image residue. In this chapter we focus on the Photo-response non-uniformity (PRNU) and its usage for source camera identification. The PRNU pattern, as we shall introduce in Section 2.3, is caused by the non-uniform response of each pixel to the same amount of incoming photons. It is systematically present in every image captured by a given sensor and it has been shown to be a sort of device fingerprint. Here, we tackle the problem of detecting the presence of a PRNU pattern in a query image and propose a new statistical certification method.

The standard procedure to estimate the PRNU of a given camera is to average the noise residuals of a certain amount of images captured by said camera [31, 129, 151]. The noise residuals are generally extracted using a denoising filter [31, 46, 136, 151]. Since these noise residuals contain other types of noise and random variations in addition to the fixed PRNU, these residuals are averaged to suppress the random variations and enhance the fixed pattern that is present in all of them. This fixed pattern can be then refined to discard non-unique artifacts that are not part of the PRNU.

Once the PRNU pattern is estimated, the problem of source camera identification consists in determining if the PRNU pattern is present in the query image or not. This problem can be stated as a statistical hypothesis testing problem, with the null hypothesis corresponding to the absence of a tested PRNU pattern, and the alternative its presence [31, 83]. The test statistic used to decide between one alternative or the other is usually a correlation metric [83, 151]. This kind of metric provides a measure of the presence of the PRNU of the camera in the given image. By comparing the observed test statistic to a pre-fixed threshold that depends on the significance level at which the test is performed, one can decide whether the query image was taken with the camera under investigation or not.

A major drawback, shared by all the test statistics proposed for PRNU detection, is that their distribution needs to be determined empirically by analyzing the behavior of a given camera's PRNU pattern with respect to images acquired using the same device and to images acquired with a different device. This poses a major problem for source camera certification since these methods do not provide an accurate false alarm rate for each detection, but rather a lower bound related to the size of the dataset used to derive such thresholds.

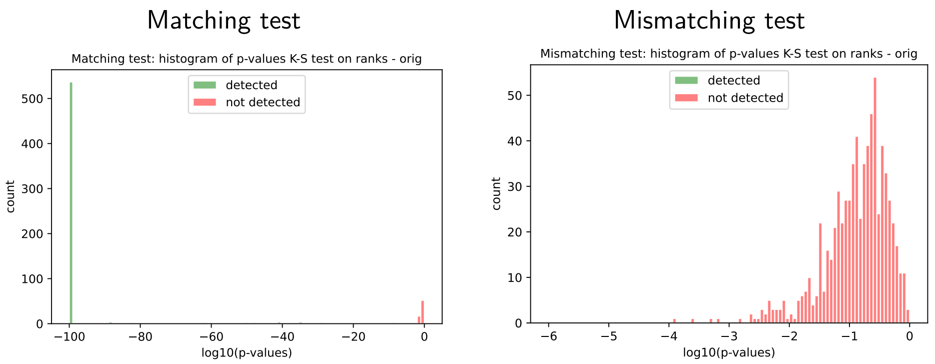


Figure 1.10: Histograms of the $\log_{10}(\text{p}-values)$ obtained with the Kolmogorov-Smirnov test on the uniformity of the ranks. The histogram in the left corresponds to the matching test and the one on the right to the mismatching test in the native resolution images from Forhheim dataset [93]. The matching histograms are truncated in -100 and, therefore, all the p-values below this bound contribute to this bin. We observe that the proposed approaches deliver very significant detections. Furthermore, the p-values obtained for the mismatching test are mostly above 10^{-1}.

In this chapter, we propose an alternative strategy for source camera certification that can be used in conjunction with the classic testing approaches. Our method relies on two hypothesis tests based on local correlations that do not require computing empirical distributions. For each detection, we provide the p-value of the test as a confidence measure. As shown in Figure 1.10, in most cases PRNU true detections are almost absolute, with p-values smaller than 10^{-100}. On the other hand, most true negatives deliver p-values above 10^{-1}.

The proposed approach complements the classic testing statistics by associating a very informative confidence measure with detections and rejections. However, our approach does not require the construction of datasets to model the test statistics under the null hypothesis. Instead, we learn a stochastic background model from the image itself. Therefore, the presented method can help certify most of the results given by the classic metrics, but without the need for empirical thresholds.

1.9 A Study of CamTE: a Camera Trace Erasing Network (Chapter 9)

So far, we have focused on the applications of noise analysis to different forensics tasks. Still, forgers may attempt to hide these traces. The following chapters shall focus on this problem.

Throughout the image formation process, the raw image acquired by an electronic sensor undergoes several operations such as denoising, demosaicing, white balance, contrast correction, and compression. These operations leave traces in the resulting image that encodes information about the camera processing chain. Such traces, though imperceptible to the naked eye, play a crucial role in various forensic tasks, as shown in the previous chapters. However, nothing prevents people from covering up these traces in order to deceive such methods. Understanding the boundaries of forensic analysis is crucial so that efforts are made to surpass them. Counter-forensics emerged as the research domain that challenges digital forensics and methodically investigates its limitations [22].

In this chapter we explore the camera trace erasing method proposed by Chen et al. [29]. This method aims at extracting the camera trace, i.e. the signal embedded in the image during the image formation process which implicitly encodes information about the camera processing chain (see Chapter 2), while preserving the image content.

Given an image I, the authors state that we can decompose it in two components: The content signal S and the camera trace T:

$$I = S + T. \tag{1.3}$$

In order to extract the camera trace T, the goal is to find $F : \mathcal{X} \to \mathcal{X}$ such that $F(I) = S$, where $\mathcal{X}$ stands for the space of all images. However, finding such F amongst all possible functions $F : \mathcal{X} \to \mathcal{X}$ is not straightforward.

The proposed camera trace erasing method is a parametric function F_θ, where θ are trainable parameters. For network training, a novel hybrid loss is designed. This hybrid loss is defined as a combination of three different losses: the embedded similarity loss, the truncated fidelity loss, and the cross-identity loss. In this section, we briefly describe the method and its results.

We qualitatively evaluate this method in three aspects: the image quality of the purified images, the direct effectiveness on JPEG trace removal, and the indirect effectiveness of deceiving the source camera model method introduced in Chapter 7. Figure 1.11 shows the ability of the camera trace erasing method to deceive the source camera model comparison method presented in Chapter 7. In addition, we offer an online demo for anyone to test their own images: `https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000443`.

The next section introduces a new counter-forensic method based on diffusion models. In Chapter 10, which we shall introduce in the next section, a quantitative evaluation of the CamTE approach is also provided.

1.10 Diffusion Models for Counter-Forensics (Chapter 10)

In this chapter, we study the capabilities of diffusion models to erase the traces left by forgers and, therefore, deceive forensics methods. Such an approach has been recently introduced for adversarial purification, achieving significant performance [175, 212].

Adversarial attacks share some common properties with image forgeries, in the sense that both techniques introduce subtle modifications to the images that, though imperceptible to the naked eye, disrupt the image's traces. The goal of adversarial attacks is to deceive a

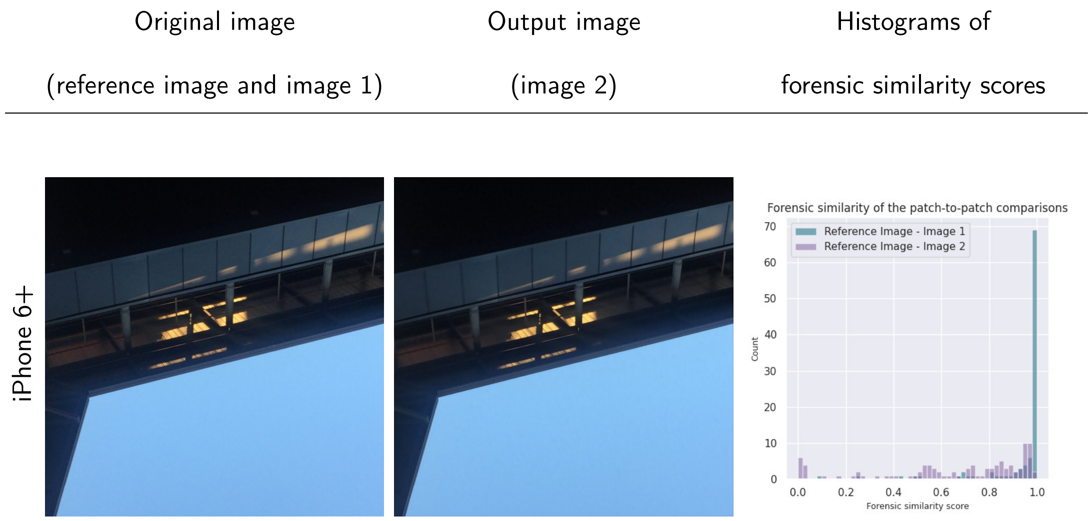


Figure 1.11: Effectiveness of the camera trace erasing method to deceive the forensic similarity approach (Chapter 7). The first figure displays the original image, which is taken as the reference image and also as image 1 in the comparison. The middle figure displays the output image, taken as image 2. Finally, the third figure shows the histogram of the forensic similarity scores obtained in the 100 patch-to-patch comparisons.

model into making incorrect predictions. Adversarial purification can be, therefore, linked to counter-forensics since it aims at pre-processing the input data to remove these adversarial perturbations. Generally, these purification methods are based on generative models [191].

In recent years, diffusion models have emerged as highly effective generative models [96, 197]. These models have showcased impressive capabilities in generating high-quality samples, outperforming traditional Generative Adversarial Networks (GANs) in the realm of image generation. The advancements in diffusion models have led to significant improvements in the fidelity and realism of synthesized images, highlighting their potential as state-of-the-art models in the field.

In this chapter we evaluate, for the first time, the efficiency of diffusion purification methods, currently used for adversarial purification [175, 212], as counter-forensics methods. The rationale behind the use of diffusion models for adversarial purification is that these models learn the distribution of clean data. Hence, by diffusing an adversarial example and then applying the reverse generative process, the diffusion model gradually removes the adversarial perturbations and reconstructs the underlying clean sample.

The same rationale can be applied to hide the forensic traces caused by tampering. Indeed, since diffusion models are trained on pristine images, diffusion purification methods applied to forged images should recover purified images without any inconsistency in the camera traces. Once such disruptions on the camera processing chain are erased, purified images should be able to deceive any forgery detection method relying on them. Figure 1.12 shows an example of the aforementioned approach: while ZERO [176] correctly detects the original forgery, once diffusion purification is applied, the method is no longer able to detect it.

The experimental analysis performed in this chapter shows that diffusion purification methods are well-suited for counter-forensics tasks. Such approaches outperform already existing counter-forensics techniques both in deceiving forensics methods, and in preserving the natural look of the purified images.

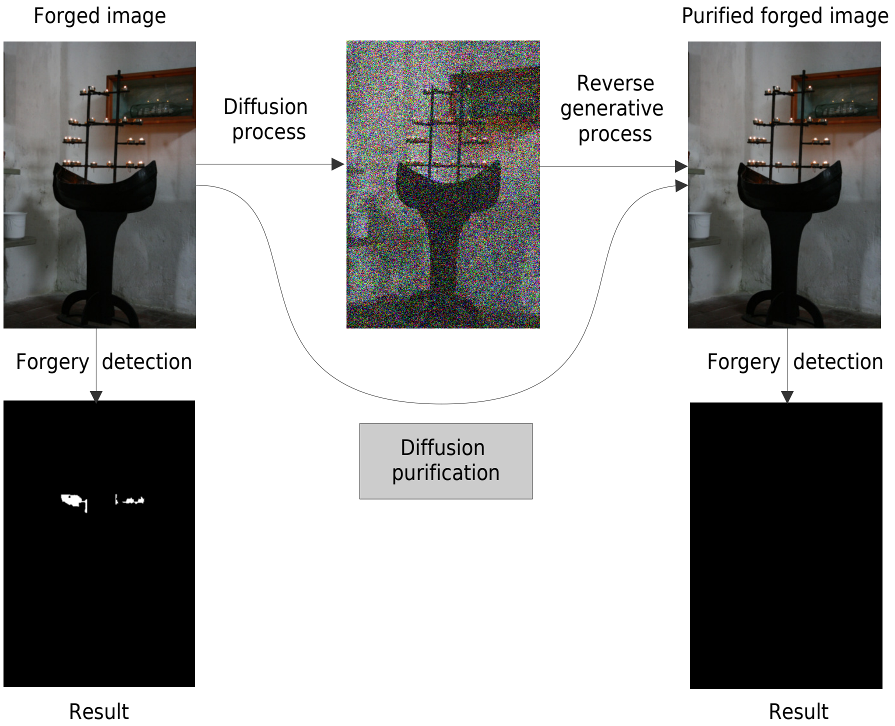


Figure 1.12: Illustration of the use of diffusion models as a counter-forensic technique. A forged image from FAU dataset [37], correctly detected by ZERO [176], produces no detection after diffusion purification.

1.11 Summary of Contributions

Heretofore, we have introduced the research work carried out during this thesis. Still, this is not the only thing researchers do. Communicating the results achieved to the concerned audience is as relevant as achieving them. Transitions in research are not mere administrative steps; they are the lifeblood of the scientific process and play a vital role in shaping the impact of research on society. They enable the accumulation of knowledge, foster collaboration, ensure ethical conduct, drive innovation, inform policy-making, and promote lifelong learning. Recognizing the importance of transitions in research is essential for researchers, policymakers, and society at large as they collectively work to address the complex challenges of the modern world and strive for progress and improvement. This thesis has led to several transitions, including publications, online demos, and real-world applications.

1.11.1 Publications

This thesis has led to the following publications:

Conference and Journal

- **Diffusion models meet image counter-forensics**, M. Tailanián, M. Gardella, A. Pardo, P. Musé. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024.* (Chapter 10)

- **SiamTE: Siamese Trace Erasing for camera trace extraction**, M. Gardella. Accepted for publicacion in *Image Processing On Line (IPOL), 2023.* (Chapter 9)

- **Image forgery detection based on noise inspection: analysis and refinement of the Noisesniffer method**, M. Gardella, P. Musé, M. Colom, J.-M. Morel. Accepted for publicacion in *Image Processing On Line (IPOL), 2023.* (Chapter 4)

- **PRNU-based source camera statistical certification**, M. Gardella, P. Musé, M. Colom, J.-M. Morel, D. Perraud. *IEEE International Workshop on Information Forensics and Security (WIFS), 2023.* (Chapter 8)

- **A Contrario Detection of H.264 Video Double Compression**, Y. Li, M. Gardella, Q. Bammey, T. Nikoukhah, R. Grompone von Gioi, M. Colom, J.-M. Morel. *IEEE International Conference on Image Processing (ICIP), 2023.* (Not included in the manuscript)

- **Détection a contrario de la double compression vidéo et application préliminaire à la détection de deepfakes**, Y. Li, M. Gardella, Q. Bammey, T. Nikoukhah, R. Grompone von Gioi, M. Colom, J.-M. Morel. *Groupe de Recherche en Traitement du Signal et des Images (GRETSI), 2023.* (Not included in the manuscript)

- **Forensic Similarity for Source Camera Model Comparison**, M. Gardella, P. Musé. *Image Processing On Line (IPOL), 2022.* Best student paper award –MLBriefs 2022. (Chapter 7)

- **Image Forgery Detection via Forensic Similarity Graphs**, M. Gardella, P. Musé. *Image Processing On Line (IPOL), 2022.* Best student paper award – MLBriefs 2022. (Chapter 5)

- **Video Signal-Dependent Noise Estimation via Inter-Frame Prediction**, Y. Li, M. Gardella, Q. Bammey, T. Nikoukhah, R. Grompone von Gioi, M. Colom, J.-M. Morel. *IEEE International Conference on Image Processing (ICIP), 2022.* (Not included in the manuscript)

- **The approach to reproducible research of the Image Processing On Line (IPOL) journal**, A. Nicolaï, Q. Bammey, M. Gardella, T. Nikoukhah, O. Boulant, I. Bargiotas, N. Monzón, C. Truong, B. Kerautret, P. Monasse, M. Colom. *Informatio, 2022.* (Chapter 1)

- **The Impact of JPEG Compression on Prior Image Noise**, M. Gardella, T. Nikoukhah, Y. Li, Q. Bammey. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022.* (Chapter 2)

- **Non-Semantic Evaluation of Image Forensics Tools: Methodology and Database**, Q. Bammey, T. Nikoukhah, M. Gardella, R. Grompone von Gioi, M. Colom, J.-M. Morel. *IEEE/CVF Winter Conference on Applications of Computer Vision, 2022.* (Chapter 6)

- **Forgery Detection in Digital Images by Multi-Scale Noise Estimation**, M. Gardella, P. Musé, J.-M Morel, M. Colom. *Journal of Imaging, 2021.* (Chapter 3)

- **NoiseSniffer: a Fully Automatic Image Forgery Detector Based on Noise Analysis**, M. Gardella, P. Musé, J.-M Morel, M. Colom. *9th IEEE International Workshop on Biometrics and Forensics (IWBF), 2021.* Best paper award – IWBF 2021. (Chapter 4)

Book chapter

- **How to Reconstruct the History of a Digital Image, and of Its Alterations, Multimedia Security 1: Authentication and Data Hiding**, Q. Bammey, M. Colom, M. Gardella, R. Grompone von Gioi, J.-M. Morel, T. Nikoukhah and D. Perraud. Directed by William Puech. *John Wiley & Sons, Inc., 2022.* (Chapter 2)

- **Comment reconstruire l'histoire d'une image digitale, et de ses altérations, Sécurité Multimédia vol. 1**, Q. Bammey, M. Colom, M. Gardella, R. Grompone von Gioi, J.-M. Morel, T. Nikoukhah and D. Perraud. Directed by William Puech. *ISTE, 2021.* (Chapter 2)

1.11.2 Reproducible research through IPOL demos

Scientific research requires that results can be accessed, tested, and replicated by other researchers. In general, performing reproducible research is not simple and it can be even impossible (for example, in the case of astrophysics replicating singular events if even out of the control of the researcher). However, in computational sciences there is no special hindrance to reproduce, repeat, and compare results. In 2009 the Image Processing On Line (IPOL) journal was founded as a modest contribution to implement reproducible research in the Image Processing field, and then expanded to more general signal-processing algorithms, such as video or physiological signal processing, among others. IPOL re-defined the concept of publication, which is no longer just the article, but the combination of the article, its source code, and any associated data needed to reproduce the results, everything as an indivisible whole. During this thesis, we developed the following IPOL demos:

- Forensic Similarity for Source Camera Model Comparison
  https://ipolcore.ipol.im/demo/clientApp/demo.html?id=424,

- Image Forgery Detection via Forensic Similarity Graphs
  https://ipolcore.ipol.im/demo/clientApp/demo.html?id=432,

- Image forgery detection based on noise inspection: analysis and refinement of the Noisesniffer method https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000341,

- SiamTE: Siamese Trace Erasing for camera trace extraction
  https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000443,

- A Signal-dependent Video Noise Estimator Via Inter-frame Signal Suppression
  https://ipolcore.ipol.im/demo/clientApp/demo.html?id=420.

1.11.3 Projects and transitions to society

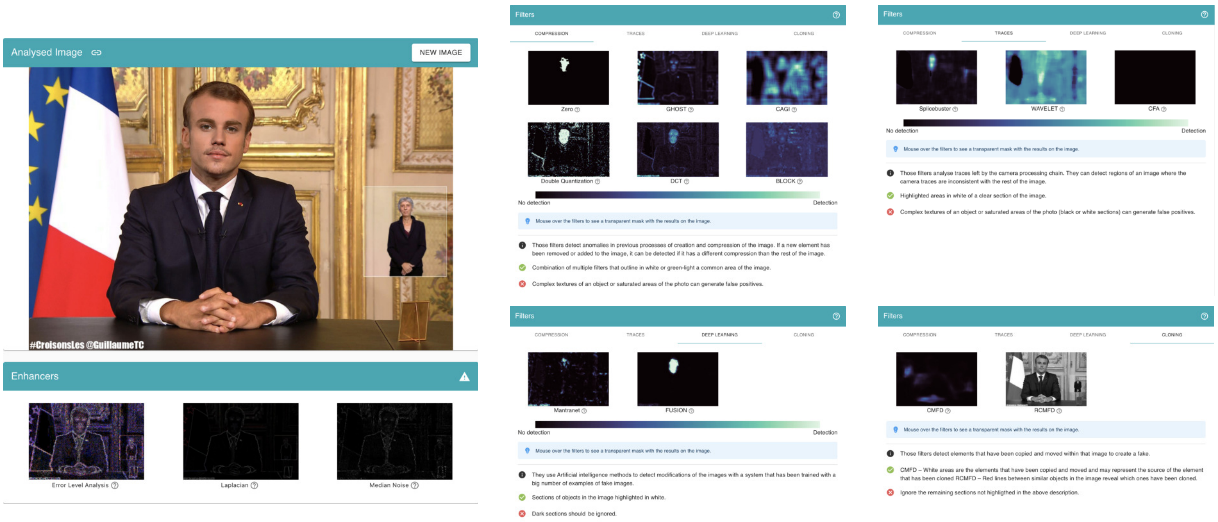
The DEFALS challenge Our societies have become both permeable and dependent on information which, thanks to the Internet, can be easily disseminated. In this context, images are a major vector of information; they are often given evidentiary value to support opinions or justify decisions. This is why they are sometimes powerful vectors of disinformation. Indeed, users of digital photographs have sophisticated means at their disposal to retouch them, or even counterfeit them, without leaving any obvious trace.

In this context, the DEFALS project [5], organized by the ANR, had the goal of initiating and advancing research in image analysis for integrity verification purposes and encouraging

[5] https://anr.fr/fr/detail/call/challenge-detection-de-falsifications-dans-des-images-defals/

connections between the image and optics communities, end users and manufacturers. This project took the form of a challenge or competition where each selected team developed an analysis system, allowing the automated detection of modifications made to images. During this project, two of the image forgery detection algorithms based on noise analysis were developed: PB (Chapter 3) and Noisesniffer (Chapter 4). These methods proved to be useful in the “into the wild” images.

EnVisu4 project: Enhanced Visual Forensics We also participated in the ENVISU4 project, financed by the International Fact-Checking Network (IFCN). The purpose of this project was to tackle forensic tools usability problems reported by fact checkers trying to debunk fake images. In response to this challenge, we significantly enhanced the forensic toolbox, with new state-of-the-art methods, completely redesigned the user interface, and integrated a new tool to compare images and export the result into an animated GIF image to better reveal image manipulation.



Mouse over the filters to see a transparent mask with the results on the image.

- Those filters detect anomalies in previous processes of creation and compression of the image. If a new element has been removed or added to the image, it can be detected if it has a different compression than the rest of the image.
- Combination of multiple filters that outline in white or green-light a common area of the image.
- Complex textures of an object or saturated areas of the photo can generate false positives.

Mouse over the filters to see a transparent mask with the results on the image.

- Those filters analyse traces left by the camera processing chain. They can detect regions of an image where the camera traces are inconsistent with the rest of the image.
- Highlighted areas in white of a clear section of the image.
- Complex textures of an object or saturated areas of the photo (black or white sections) can generate false positives.

Mouse over the filters to see a transparent mask with the results on the image.

- They use Artificial intelligence methods to detect modifications of the images with a system that has been trained with a big number of examples of fake images.
- Sections of objects in the image highlighted in white.
- Dark sections should be ignored.

Mouse over the filters to see a transparent mask with the results on the image.

- Those filters detect elements that have been copied and moved within that image to create a fake.
- CMFD – White areas are the elements that have been copied and moved and may represent the source of the element that has been cloned RCMFD – Red lines between similar objects in the image reveal which ones have been cloned.
- Ignore the remaining sections not highligthed in the above description.

Figure 1.13: Preview of the new version of the platform. The different forensic filters are applied to an image from satirical photoshopper @GuillaumeTC on Twitter.

Figure 1.13 showcases the new version of the platform, featuring new state-of-the-art methods classified according to the inconsistencies they search for and a new tab displaying image enhancers.

VERA.AI project: VERification Assisted by Artificial Intelligence Together with 14 partners, we are currently part of VERA.AI project, funded by EU Horizon Europe, the UK’s innovation agency and the Swiss State Secretariat for Education, Research and Innovation (SERI). It is the aim of the VERA.AI project and its partners to develop and build trustworthy AI (Artificial Intelligence) solutions in the fight against disinformation. These are to be co-created in close collaboration between leading technology experts in the domain and prospective future users–all brought together in the VERA.AI project consortium, following a multidisciplinary co-creation approach.

All this is to deliver solutions that can be used by the widest possible community such as journalists, investigators, researchers and such like, while also setting the foundations for future research and development in the area Artificial Intelligence against disinformation. The expected solutions will deal with different content types (audio, video, images, and text) and do so across a variety of languages. They will mostly be open and accessible to and usable by anyone.

APATE project: A Prototype deepfake Assessment Toolbox for forensic Experts We are currently participating in the APATE project, funded by the ANR, together with SNPS (National Scientific Police Service), LIX (Ecole Polytechnique), EPITA (School for Computer Science and Advanced Techniques) and IDM (Idemia I&S). The goal is to advance the state of the art on methods of detection of deepfakes, with the necessary environment to make them usable by experts in the courts.

Esta página ha sido intencionalmente dejada en blanco.

Chapter 2

The camera processing pipeline and its traces on noise

This chapter provides the necessary background that this thesis builds upon. We first briefly describe the image processing pipeline and the traces each of the operation leave on the noise model. We then give a more in-depth analysis of the effects of JPEG compression on prior image noise. As we shall see, it is this step that modifies the most the noise model, which initially follows a Poisson distribution. Finally, we propose an overview of the use of the traces left by the camera processing chain for forgery detection. Parts of the three first sections of this chapter have been published as a chapter of a book [12, 183]. The work presented in the fourth section is published as *The Impact of JPEG Compression on Prior Image Noise* in the ICASSP conference [74] and an online demo is available at: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000136.

2.1 Introduction

The goal of this chapter is to briefly review each step of the processing chain of an image, how they affect the noise present in images and how the traces left can be used for forgery detection. This information can sometimes appear in the data accompanying the image, called EXIF (*Exchangeable Image File Format*), which also includes information such as the brand and model of the camera and lens, the date and location of the photograph, and its shooting settings. However, this data can be easily modified, and is often automatically deleted by social media for privacy reasons [80]. Therefore, we are interested in the information left by the operations on the image itself and not in the metadata. Some methods, like the one presented in [99], offer to check the consistency of the data present in the image with its EXIF metadata.

The knowledge of the image processing chain allows for the detection of the presence of certain changes. A first application is the authentication of the model of a camera. The processing chain is specific to each device model; so it is possible to determine the device model by identifying the processing chain [207]. This subject will be covered in Part II. A second series of methods are based on the study of the residue – sometimes called noise – left by the processing chain. This residue is made up of all the traces left by each operation. Using this idea, Cozzolino and Verdoliva proposed to use steganography tools to extract the image residue [49]. Treating this residue as a hidden information in the image, an algorithm such as Expectation-Maximization (EM) is then used to classify the different regions of the image. Subsequently, neural networks have shown good performance in extracting the residue automatically [48, 78], or even in carrying out the classification themselves [240]. While it is often difficult, or even impossible, to distinguish each step in the processing chain individually, it is easier to distinguish two different processing chains as a whole. This idea is exploited by O. Mayer and M.C. Stamm [162] to construct a forensics traces similarity score. Such an approach will be covered in Chapter 5.

The outline of this chapter stems from previous considerations. Before getting into the problem ourselves, we proceed in Section 2.2 with a description of the main operations of the image processing chain. Section 2.3 is dedicated to the transformations that the noise of the raw image undergoes at each operation applied during a standard processing chain leading to the visible image. We then dedicate Section 2.4 to a more in depth study of the impact of JPEG compression on prior image noise. Indeed, as we shall see in Section 2.3, JPEG compression is the operation that has the most impact on noise. Finally, Section 2.5 will briefly review how the traces left by the camera processing pipeline can be used to detect forgeries.

2.2 Describing the image processing chain

The main steps in the digital image acquisition process are illustrated in Figure 2.1. Together with the corresponding image at each step, we provide a zoom in on the details of each image, so that the reader can spot the fine details introduced by each operation.

2.2.1 Raw image acquisition

The first step of acquiring a raw image consists of counting the number of incident photons over the sensor along the exposure time. There are two different technologies used in camera sensors: Charge Coupled Devices (CCDs) and Complementary Metal-Oxide-Semiconductors (CMOS). Although their operating principles differ, both can be modelled in a very similar way [1]. Both sensors transform incoming light photons into electronic charge which interacts with detection devices to produce electrons stored in a potential light well. When the latter is full, the pixels become saturated, and the electrons are no longer as into output voltage values. The final step is to convert the analog voltage measurements into digital quantized values.

2.2.2 Demosaicing

Most cameras cannot see color directly, because each pixel is obtained through a single sensor which can only count the number of photons reaching it in a certain wavelength range. In order to obtain a color image, a color filter array (CFA) is placed in front of the sensors. Each of them only counts the photons of a certain wavelength. As a result, each pixel has a value relative to one color. By using filters of different colors on neighbouring pixels, the missing colors can then be interpolated.

Although other exist, almost all cameras use the same CFA: the Bayer array, which is illustrated in Figure 2.2. This matrix samples half the pixels in green, a quarter in red, and the last quarter in blue. Sampling more pixels in green is justified by the human visual system, which is more sensitive to the green color.

Unlike other steps in creating an image, a wide variety of algorithms are used to demosaic an image. Bilinear interpolation is the simplest of the demosaicing algorithms. It consists of linear interpolation of missing colors by the average of a pixel's direct neighbours sampled in that color. This method is simple, but tends to produce strong aberrations in non-flat regions, especially in the presence of edges or details.

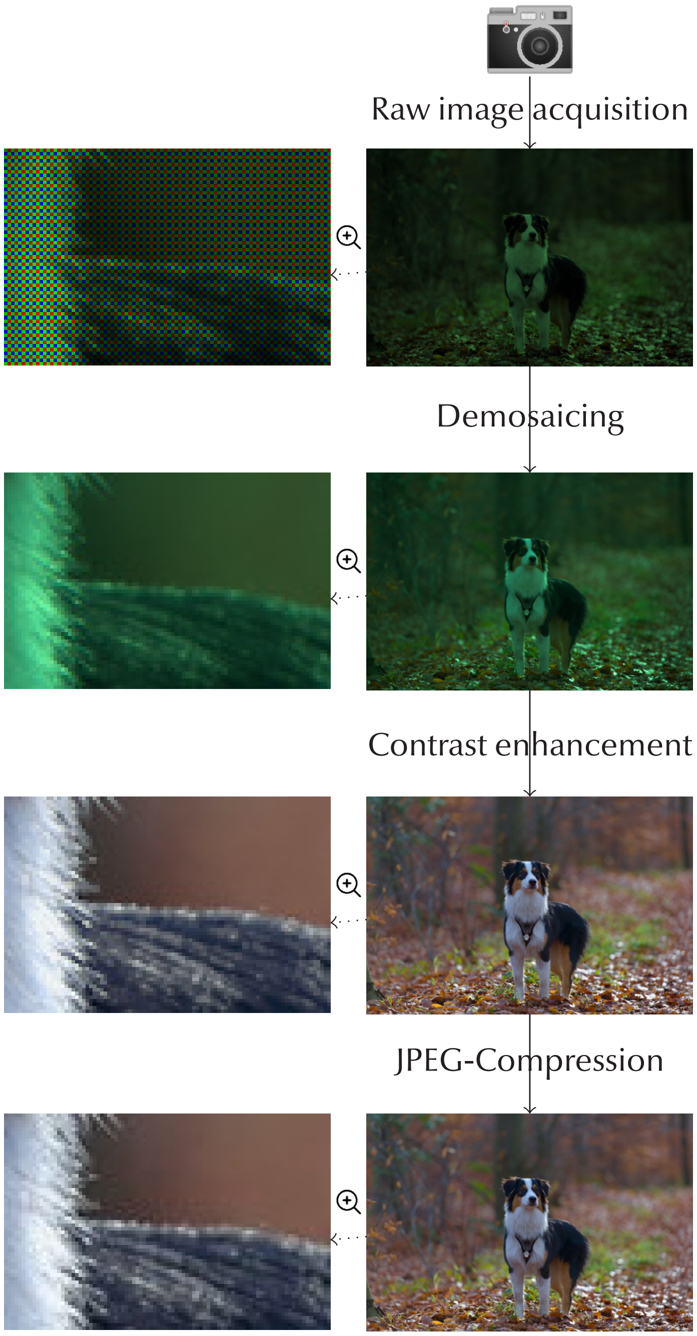


Figure 2.1: Simplified processing pipeline of an image, from its acquisition by the camera sensor to its storage as a JPEG-compressed image. The right column represents the image as it goes through each step. The left column shows the details of the image obtained at each step.

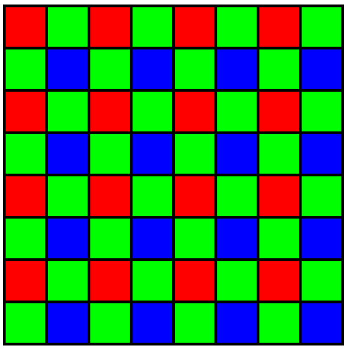


Figure 2.2: The Bayer Matrix is by far the most used for sampling colors in cameras.

To avoid these artefacts, more recent methods attempt to simultaneously take into account information from the three color channels and avoid interpolating along a steep gradient. For instance, the Hamilton-Adams method is carried out in three stages [94]. First, it interpolates the missing green values by taking into account the green gradients corrected for the discrete Laplacian of the color already known at each pixel to interpolate horizontally or vertically, in the direction where the gradient is weakest. It then interpolates the red and blue channels on the pixels sampled in green, taking the average of the two neighbouring pixels of the same color, corrected for the discrete Laplacian of green in the same direction. Finally, it interpolates the red channel of blue-sampled pixels and the blue channel of red-sampled pixels using the corrected average of the Laplacian of the green, in the smoothest diagonal.

Linear Minimum Mean-Square Error demosaicing [76] suggests working not directly on the three color channels (red, green and blue), but on the pixelwise differences between red and green on the one hand, and between blue and green on the other. It interpolates this difference separately in the horizontal and vertical directions, in order to estimate first the green channel, followed by the differences between red and green, and then between blue and green. The red and blue channels can then be recovered by a simple substraction. This method, as well as many other, makes the underlying assumption that the difference of color channels is smoother than the color channels themselves, and therefore easier to interpolate.

More recently, convolutional neural networks have been proposed to demosaic an image. For instance, Demosaicnet uses a convolutional neural network to jointly interpolate and denoise an image [60, 77]. Even if these methods offer superior results to algorithms without training, they also require more resources, and are therefore not widely used yet in digital cameras.

The methods described here are only a brief overview of the large array of methods that exist for image demosaicing. This variety is increased by the fact that most industrial cameras do not disclose their demosaicing algorithm, which is often private.

No demosaicing method is perfect – after all, it is a matter of reconstructing missing information – and produce some level of artefacts, although some produce much fewer artefacts than others. Therefore, it is possible to detect these artefacts to obtain information on the demosaicing method applied to the image.

2.2.3 Color Correction

White balance aims to adjust values obtained by the sensors so that they match the colors perceived by the observer by adjusting the gain values of each channel. The way in which white balance adjusts the output depends on the characteristics of the light sources, and is done so that achromatic objects from the real scene are rendered as such [149]. For example, white balance can be achieved by multiplying the value of each channel, so that a pixel that has a maximum value in each channel is found to have the same maximum value 255 in all channels.

Then, the image goes through what is known as gamma correction. The charge accumulated by the sensor is proportional to the number of photons incident on the device during the exposure time. However, human perception is not linear with the signal intensity [65]. Therefore, the image is processed to accurately represent human vision by applying a concave function of the form $f_{k,\gamma} = ku^{\frac{1}{\gamma}}$, where γ typically varies between 1.8 and 2.2. The idea behind this procedure is not only to enhance the contrast of the image, but also to encode more precisely the information in the dark areas, which are too dark in the raw image.

Nevertheless, commercial cameras generally do not apply this simple function, but rather a tone curve. Tone curves allow image intensities to be mapped according to pre-computed tables that simulate the non-linearity present in human vision.

2.2.4 JPEG compression

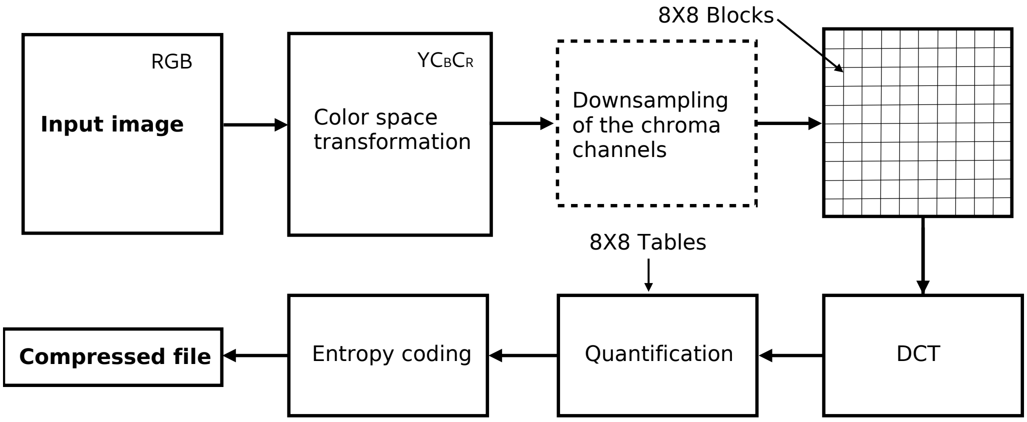


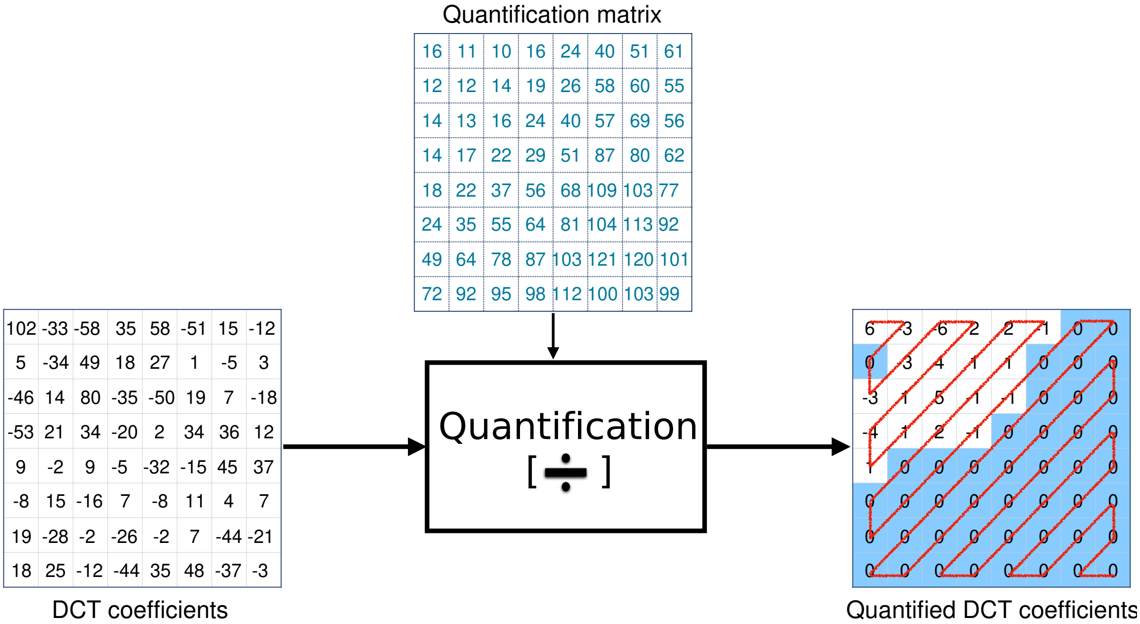
Figure 2.3: JPEG compression pipeline.

The stages of the JPEG compression algorithm, illustrated in Figure 2.3, are detailed below. The first stage of the JPEG encoding process consists of performing a color space transformation from RGB to $\mathrm{YC_BC_R}$ where Y is the luminance component and $\mathrm{C_B}$ and $\mathrm{C_R}$ are the chrominance components of the blue difference and the red difference. Since HVS is less sensitive to color changes than to changes in luminance, color components can be subsampled without affecting visual perception too much. The subsampling ratio generally applied is 4:2:0, which means that the horizontal and vertical resolution is reduced by a factor of 2. After the color subsampling, each channel is divided in blocks of 8×8 and each block is processed independently. The Discrete Cosine Transform (DCT) is applied to each block and the coefficients are quantized.

The JPEG quality factor Q, ranging between 1 and 100, corresponds to the rate of image compression. The lower this rate, the lighter the resulting file, but the more deteriorated the image. A quantization matrix linked to Q provides a factor for each component of the DCT blocks. It is during this quantization step that the greatest loss of information occurs, but it is also this step that allows the most space in memory to be saved. The coefficients corresponding to the high frequencies, of which the HVS struggles to distinguish the variations, are the most quantized, sometimes going so far as to be entirely cancelled.

Finally, as in the example in Figure 2.4, the quantized blocks are encoded without loss to obtain a JPEG file. Each 8×8 block is zig-zagged and the coefficients are arranged in the form of a vector in which the first components represent the low frequencies and the last ones represent the high frequencies.

Lossless compression by RLE coding (range coding) then exploits the long series of zeros at the end of each vector due to the strong quantization of the high frequencies, and then a Huffman code allows for a final lossless compression of the data, to which a header is finally added to form the file.



Quantification matrix

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

102	-33	-58	35	58	-51	15	-12
5	-34	49	18	27	1	-5	3
-46	14	80	-35	-50	19	7	-18
-53	21	34	-20	2	34	36	12
9	-2	9	-5	-32	-15	45	37
-8	15	-16	7	-8	11	4	7
19	-28	-2	-26	-2	7	-44	-21
18	25	-12	-44	35	48	-37	-3

DCT coefficients

6	-3	-6	2	2	-1	0	0
0	-3	4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantified DCT coefficients

Figure 2.4: An example of the impact of quantization on a DCT block. Each DCT coefficient is quantized by a value found in a quantization matrix. Rounding to the nearest integer results in many of the high frequency coefficients being set to zero. Each block is zig-zagged to be encoded as a vector with a sequence of zeros.

2.3 The avatars of noise throughout the camera processing chain

This section examines the way in which noise is affected at each step of the camera processing chain, described in the previous section. To do so, we estimate the noise curves of each of the intermediate images generated along the camera pipeline using the extended Ponomarenko et al. method [41]. This methods estimates, for each image intensity, the standard deviation of the noise associated to this value. The estimated noise curves along the camera processing chain are displayed in Figure 2.5.

Raw image acquisition

The value at each pixel generated by the process described in Section 2.2.1 can be modelled as a Poisson variable, whose expectation is the real noiseless value of the pixel. The noise measured at the CCD or CMOS sensor has several components; Table 2.1 describes the main sources.

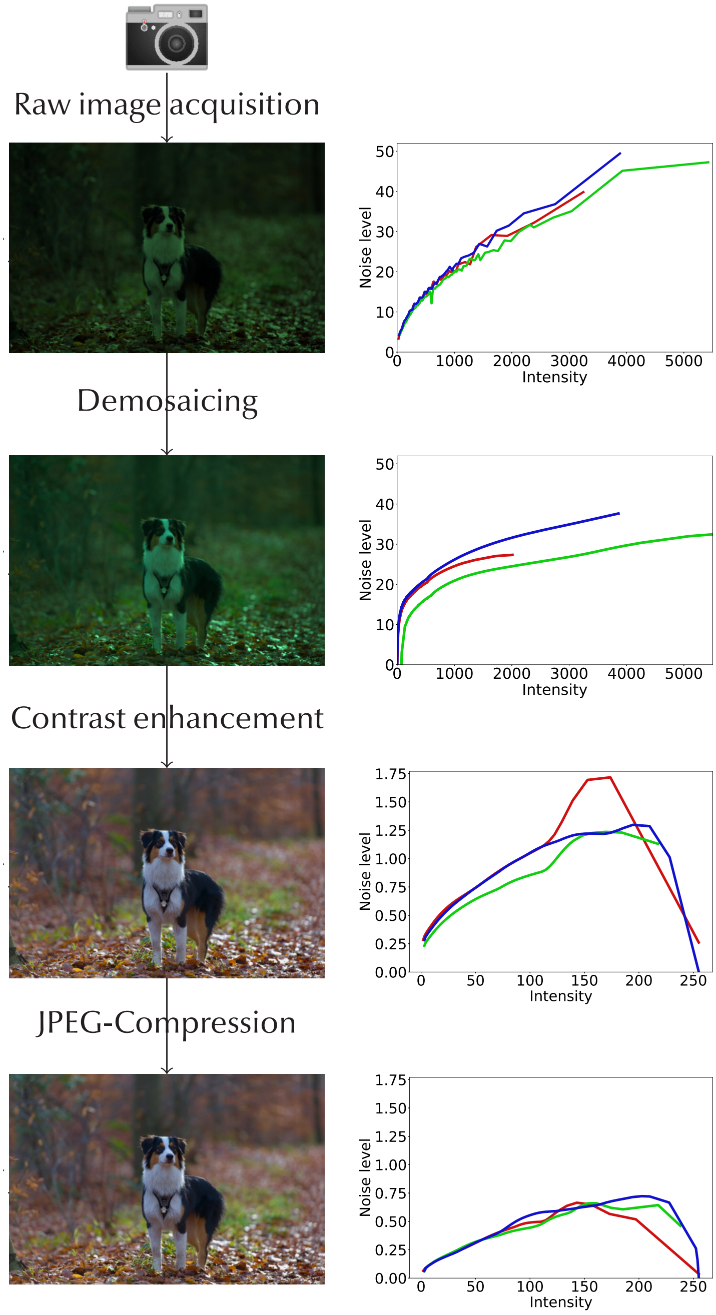


Figure 2.5: Simplified processing pipeline of an image, from its acquisition by the camera sensor to its storage as a JPEG-compressed image. The left column represents the image as it goes through each step. The right column plots the noise of the image as a function of intensity in all three channels (red, green blue) [41].

Figure 2.5 shows that, at this step, all channels have the same noise curve. As noise follows a Poisson distribution, the noise variance follows a simple linear relation $\sigma^2 = a + bu$, where u is the intensity of the ideal noiseless image, and a and b are constants. Consequently, the noise curves are strictly increasing. Moreover, although the noise curves do not account for it, the noise characteristics reported above suggest that the noise is uncorrelated, i.e. the noise at a certain pixel is not related to noise at any other pixel with the same signal intensity.

Type of noise	**Description**
Shot noise	Due to the physical nature of light. It describes the fluctuations in the number of photons detected due to their independent emission from each other.
Dark noise	Some electrons accumulate on the potential well as the result of a thermal cause. These electrons are known as dark current because they are present and will be detected even in the absence of light.
Photo Response Non-Uniformity (PRNU)	It describes the way in which the individual pixels in the sensor array respond to uniform light sources. Due to variations in pixel geometry, substrate material, and micro-lenses, different pixels do not produce the same number of electrons from the same number of photons hitting them.
Readout noise	During the readout phase of the acquisition process, a voltage value is read at each pixel. This voltage is computed as a potential difference from a reference level which represents the absence of light. Thermal noise, inherent in the readout circuit, affects the output values.
Electronic noise	It is caused by the absorption of electromagnetic energy by the semiconductors of the camera circuits and the cross-talk phenomenon.

Table 2.1: Description of the main sources of noise during the acquisition process

Demosaicing

Demosaicing is presented in more detail in Section 2.2.2. After this step, the noise at each pixel is correlated with its neighbours. After demosaicing each channel has a different noise curve since channels are processed differently by the demosaicing algorithm.

Color correction

White balance increases the intensity range of the image. Since the weights are different in each color channels, as mentioned in Section 2.2.3, the three noise curves are less correlated after this step. Then, gamma correction greatly increases the noise and the dynamic range of the image, due to the power law function. Furthermore, the noise curves are no longer monotonically increasing after this step. Indeed, if we denote γ the function applied during the gamma correction step, the asymptotic expansion around the intensity u yields $\gamma(u+n) = \gamma(u)+\gamma'(u)n$ where n is the noise at the intensity u.

JPEG Compression

The dynamic range remains unchanged after JPEG compression. However, noise is reduced after JPEG compression due to the quantization of the DCT coefficients, in particular those corresponding to high frequencies. In this case, the curve estimated by the Ponomarenko et al. algorithm [41] shall not be accurate since the noise estimation method estimates noise at high frequencies, which are altered or even destroyed by compression.

The noise present in JPEG images is the result of several transformations on the initial noise model, which initially follows a Poisson distribution. In the end, the final image's noise does not follow any predefined model, it rather depends on many unknown parameters which are set by each manufacturer. The only certainty we have is that noise is intensity-dependent and frequency-dependent.

2.4 The Impact of JPEG Compression on Prior Image Noise

JPEG compression is widely used to store digital images and extensive studies analysed its impact on the image quality; in particular the quantization noise and artefacts created by JPEG. Nevertheless, there is little work on the impact of JPEG compression on the noise already present in the image, which we briefly introduced in the previous section. In this section, we propose a model predicting how the noise power is affected by JPEG compression. This allows for a better understanding the noise traces on the image, which is crucial for image forensic analysis and image restoration. An interactive demo for this section is available at https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000136.

2.4.1 Modeling compressed noise

From the moment a raw image is acquired until the final JPEG picture is obtained, a complex processing chain is applied. Each of these operations alters the noise model. Indeed, the initial Poisson-Gaussian noise [67] undergoes several operations resulting in a complex noise model, as reviewed in the previous section. The final stage in most digital images consists in JPEG compression. Indeed, to be stored or transferred in a reasonable amount of time, images must undergo a compression step.

Many noise estimation algorithms [59, 178, 179, 226] suppose that noise can be estimated using only the high frequency coefficients of small patches in an image. However, this is not true for JPEG-compressed images, since the quantization step during the compression process attenuates these high frequencies. Indeed, for JPEG images, noise variance decreases as the frequency increases. As a result, these noise estimation methods yield to an inaccurate estimation of the noise. Noise estimation is a mandatory step of countless image processing tasks such as denoising [51, 235], forgery detection [49, 73], anomaly detection [61], PRNU

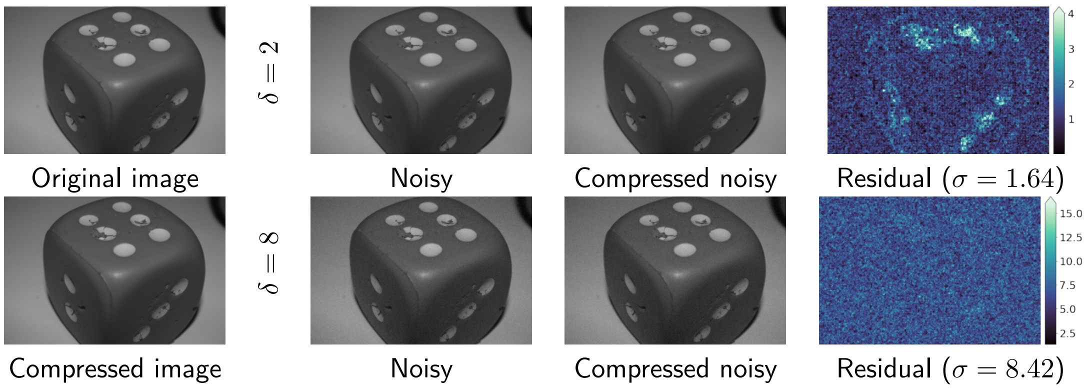


Figure 2.6: Gaussian noise is added to the `dice` image before compression. An estimation of the remaining noise is obtained by computing the difference between the noisy and noiseless images, both after compression. This enables us to estimate specifically the effect of the compression on existing noise, while ignoring most of the noise coming directly from the compression. As per our model, a low noise level is diminished even more by the compression, whereas a higher noise level is instead augmented. Compression is done with the Pillow library at a quality factor of 85.

extraction [31, 186] and steganography [200, 201], just to mention a few. The performance achieved by the methods developed to tackle each of these tasks depend on how accurately they are able to estimate noise.

The aim of this section is to provide a characterisation of the resulting noise after JPEG compression. Such a model could help accurately estimate noise, boosting the performance of a huge variety of image processing tasks that require noise estimation as one of their fundamental steps.

After JPEG compression, there are two kinds of noise in the image: the original noise, that was there before compression and was compressed with the rest of the image, and the noise coming from the JPEG compression itself. There is consequent literature on estimating the noise that directly comes from compression itself [95, 134, 135]. Due to the lossy nature of JPEG, it affects tasks in various domains. In image forensics, subtle traces such as demosaicing artefacts are much harder to analyse when the image is compressed [130]. Mandelli *et al.* [157] analyse the effects of JPEG compression on different camera traces and show that care must be taken from the training data to ensure some robustness to compression. Image classification, especially when done at full resolution, is also affected by compression [128].

In comparison, analysis of how the compression affects already-present noise has received little attention. We propose to study in details how JPEG compression affects noise that was present prior to the compression. Indeed, noise is present in an image from the sensing of the real scene. Each step in the pipeline, including JPEG compression, alters the already-existing noise in some way, while often adding its own noise on top of it. See [107] for a general overview of the impact of each processing step on the noise. Jiang *et al.* [105] notice that counter-forensics techniques, aimed at hiding JPEG compression, usually introduce inconsistent noise in the image. Noise-level analysis is applied to distinguish authentic regions from region with hidden traces of a previous compression. Corchs *et al.* [45] study the influence of images distortions, in particular Gaussian noise and JPEG compression, on the overall image quality. While this study combines Gaussian noise that is subjected to further JPEG compression, they do not directly study the influence JPEG compression had on the noise, but rather the combined effect both had on the end image's quality.

2.4.2 JPEG Compression on Gaussian Noise

Although the noise present in an image before JPEG compression is not Gaussian, the Poisson distribution of noise derived from the photon count can be approximated by a Gaussian distribution. However, this Gaussian distribution would still be intensity dependent. In order to transform this signal dependent noise into homoscedastic, a variance stabilizing transformation can be applied [6]. This procedure allows us to work with the popular white Gaussian noise assumption used in many noise estimation and denoising algorithms.

Let n be the spatially-independent zero-mean Gaussian noise existing in an image before JPEG compression. We note by $\tilde{u} \triangleq u + n$ the noisy observation in pixels, were u is the ideal noiseless image. Here we focus on the luminance colour space which is not downsampled and thus preserves more image details. Excluding the effect of lossless compression ($QF = 100$), the impact of lossy JPEG compression can be modelled as follows:

Consider an 8×8 block, the first operation consists in performing the DCT transformation of the block. Then, the DCT coefficients are converted into integers with division by a quantization table, followed by rounding. During the decoding stage, the multiplication by the quantization table is performed, followed by the inverse DCT (IDCT). The whole JPEG encoding-decoding process can be expressed as:

$$\tilde{u}' = \mathrm{IDCT} \circ \mathrm{Quant} \circ \mathrm{DCT}\left(\tilde{u}\right) = \mathbf{A}^{\mathsf{T}} \left[\mathbf{A}\tilde{u} \odot \frac{1}{q}\right] \odot q, \tag{2.1}$$

where $\mathbf{A} \in O(64, \mathbb{R})$ is the orthogonal matrix of DCT, $[\cdot]$ represents the rounding operation, $q \in \mathbb{R}^{64}_{+}$ is the quantization table and $\odot$ denotes the element-wise multiplication.

Since n is Gaussian, the noisy observation $\tilde{u}$ is in fact a Gaussian vector. Furthermore, if we assume that for all the pixels in a patch, the noise level is constant and equal to σ then $\tilde{u} \sim \mathcal{N}\left(u, \Sigma_{\tilde{u}}\right)$, with $\Sigma_{\tilde{u}} = \sigma^2 I.$ The DCT is an orthogonal transformation and, therefore,

$$x \triangleq DCT\left(\tilde{u}\right) \sim \mathcal{N}\left(Au, \Sigma_x\right), \tag{2.2}$$

where $\Sigma_x = \Sigma_u = \sigma^2 I.$

The input of $[\cdot]$ in the pipeline, noted as $z \triangleq x \odot \frac{1}{q}$ is still a Gaussian vector: $z \sim \mathcal{N}\left(\mu_z, \Sigma_z\right)$ with $\mu_z = \mathbf{A}u \odot 1/q$ and $\Sigma_z = diag\left(\sigma^2/q_1^2, \ldots, \sigma^2/q_{64}^2\right)$. However, the impact of the rounding operation $[\cdot]$, described in detail in [216], is not linear. The rounded z is $z' \triangleq [z]$ has expected value for each entry

$$\mathbb{E}[z_i'] = \mu_{z,i} + \frac{1}{\pi} \sum_{l=1}^{\infty} e^{\left(-\frac{2(\sigma\pi l)^2}{q_i^2}\right)} \sin\left(2\pi l \mu_{z,i}\right) \frac{(-1)^l}{l}. \tag{2.3}$$

Compared to textured patches, evaluating noise on flat patches is easier since the signal is uniform and the variation only contains noise. Thus we focus on the study of $\tilde{u} \leftarrow \tilde{u} - \mathrm{mean}\left(\tilde{u}\right)$ for the pre-compression image and $\tilde{u}' \leftarrow \tilde{u}' - \mathrm{mean}\left(\tilde{u}'\right)$ for the post-compression image. Then we get $\tilde{u} \sim \mathcal{N}\left(0, \sigma^2 I\right)$ and $x \sim \mathcal{N}\left(0, \sigma^2 I\right)$. The input of $\mathrm{Quant}\left(\cdot\right)$ is $z \sim \mathcal{N}\left(0, diag\left(\sigma^2/q_1^2, \ldots, \sigma^2/q_{64}^2\right)\right)$ which has zero mean. The rounded zero-mean Gaussian variable has a zero expectation $\mathbb{E}\left[z_i'\right] = \mathbb{E}\left[z_i\right] = 0$ according to Equation 2.3, and its variance is

$$\mathrm{Var}\left(z_i'\right) = \frac{\sigma^2}{q_i^2} + \frac{1}{12} + \sum_{l=1}^{\infty} (-1)^l \left(\frac{1}{(\pi l)^2} + \frac{4\sigma^2}{q_i^2}\right) e^{-\frac{2(\sigma\pi l)^2}{q_i^2}}. \tag{2.4}$$

Finally, the variation after compression $\tilde{u}_i' = \mathbf{A}^{\mathsf{T}}_{:,i} z' \times q_i$ satisfies $\mathbb{E}\left(\tilde{u}_i'\right) = 0.$ If we define

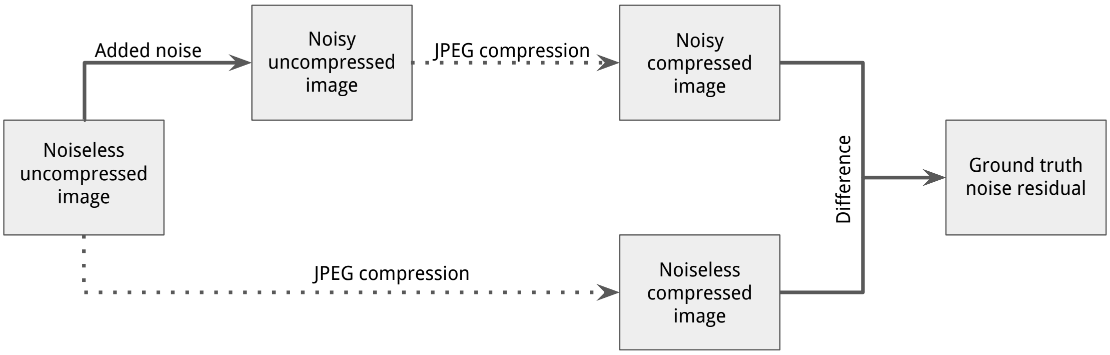


Figure 2.7: Ground truth residual computation from noiseless uncompressed images.

${\sigma'_j}^2 \triangleq \mathrm{Var}\left(\tilde{u}'_j\right)$, we get

$$ {\sigma'_j}^2 = \sum_{i=1}^{64} \left(\mathbf{A}_{i,j}\right)^2 \mathrm{Var}\left(z'_i\right) q_i^2. \tag{2.5} $$

In particular, if the quantization table satisfies $q_i = q \ \forall i$ then,

$$ {\sigma'_j}^2 = \sum_{i=1}^{64} \left(\mathbf{A}_{i,j}\right)^2 \mathrm{Var}\left(z'\right) q^2 = q^2 \mathrm{Var}\left(z'\right) \triangleq {\sigma'}^2, \tag{2.6} $$

which is independent of the pixel index j. We get the relation between the standard deviation σ of the original noise and σ' that of the quantized noise:

$$ \frac{\sigma'}{q} = \sqrt{\frac{\sigma^2}{q^2} + \frac{1}{12} + \sum_{l=1}^{\infty} (-1)^l \left(\frac{1}{(\pi l)^2} + \frac{4\sigma^2}{q^2}\right) e^{-\frac{2(\sigma \pi l)^2}{q^2}}}. \tag{2.7} $$

2.4.3 Experiments

To validate the model derived in Section 2.4.2 we conducted several experiments on both synthetic noise images and real noiseless images to which we added noise. The Noise-Free Test Images dataset [39] contains 16 high-qualities images that were carefully downsampled to remove traces of previous noise, allowing us to control the amount of noise to add to the image.

To compute the ground truth noise residuals we first select an uncompressed noiseless image (flat or from the dataset). Given a quality factor QF and a noise level σ, the noiseless uncompressed image is processed in two different ways. The first one consists in adding white noise of variance σ^2 to the uncompressed noiseless image and then compressing it with quality factor QF. The second consists only in performing the compression step, without adding noise. The ground truth noise residual is computed as the difference between the noisy compressed image and the noiseless compressed image. This computation is summarised in Figure 2.7.

Figure 2.8 shows the standard deviation of the noise residual for each 8×8 DCT coefficient and for different compression qualities, and for a fixed pre-compression noise level equal to 10. The noise's standard deviation decreases as the frequency increases, as suggested by our model. However, this effect is not homogeneous across different JPEG qualities, we can notice that the lower the JPEG quality, the more notorious this effect becomes. It is also remarkable that lowering the JPEG quality more strongly damages the high frequencies of noise, while keeping

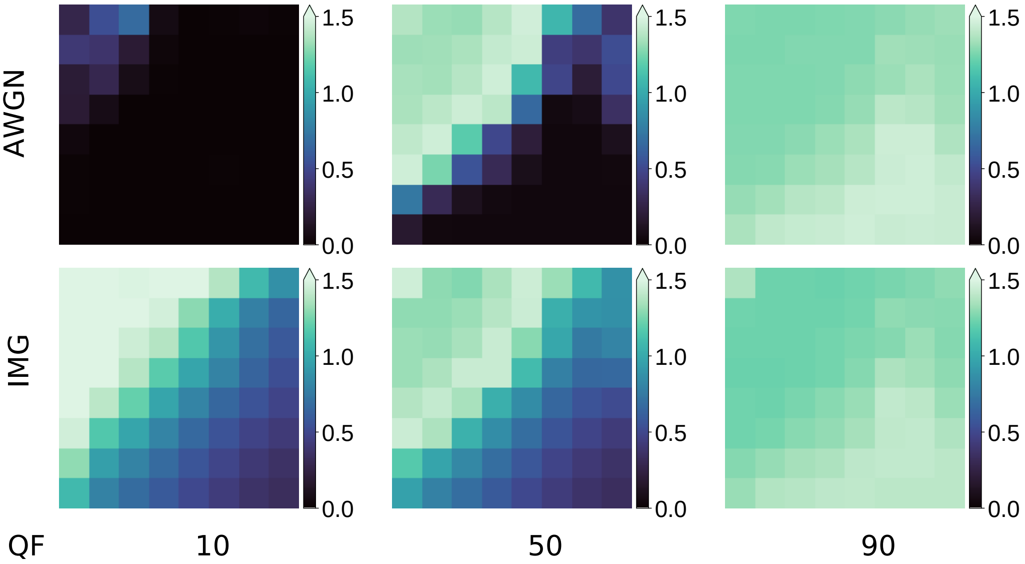


Figure 2.8: Noise residual standard deviation for each DCT coefficient, for quality factors $QF = 10, 50, 90$ and for a fixed pre-compression noise $\sigma = 10$. The top row corresponds to synthetic Gaussian noise (constructed by taking the noiseless uncompressed image as a completely flat image) and the bottom row to an image from the NFTI dataset with added noise [39].

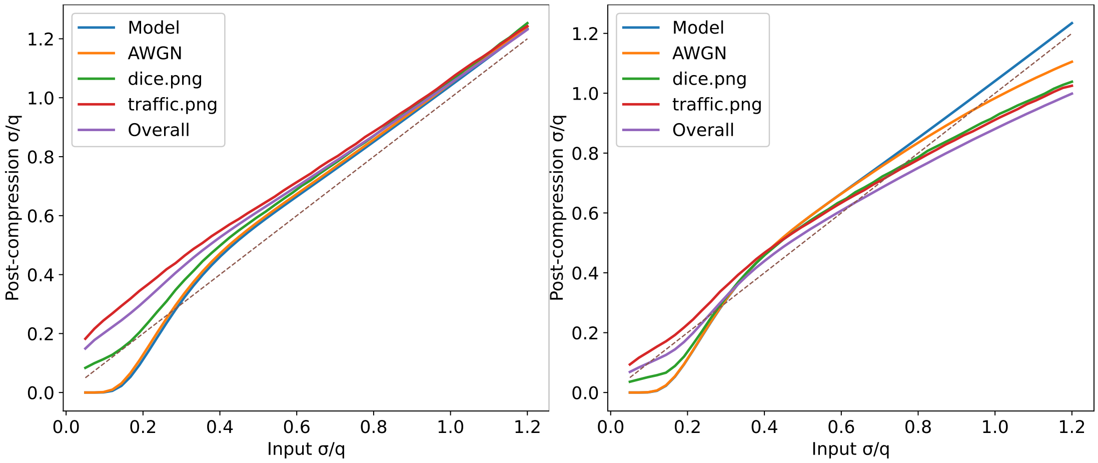


Figure 2.9: Ratio of the post-compression noise standard deviation and quality factor $\frac{\sigma'}{q}$ as function of the same ratio before compression, with $\sigma \in [1, 20]$ and $q = 8$ (left), $q = 64$ (right). The model curve corresponds to Equation 2.7 and highly coincides with the pure noise curve.

the low ones mostly unchanged. Furthermore, for a compression quality $QF = 90$ we observe that the effects of compression are pretty innocuous as the noise standard deviation is still mostly homogeneous across frequencies.

Figure 2.9 shows the standard deviation of the noise after JPEG compression divided by q as a function of the pre-compression noise divided by q, for different input images and quantization factors $q = 8, 64$. We observe that for very small values of noise, the output noise is higher than the input noise on real images. Although the theory suggests output noises should be smaller in this range of values, the experimental results are disturbed by JPEG noise. This JPEG noise is negligible when compared to higher noise levels but becomes predominant when the noise levels are very low. For medium noise levels we observe, as suggested by our model, that output noises are smaller than input noises. This is due to the effect of quantization since it removes the small variations in pixel's values for which noise is responsible. As the noise grows, we observe that the curves converge to the identity when q is low enough, as predicted by the model on pure noise. This is explained by the fact that, for large enough noise levels, the quantization factor becomes negligible with respect to the noise level. However, with a larger value of q, the noise seems dimmed by compression, deviating from the theoretical model. In practice, this is only caused by clipping; indeed to reach a $\frac{\sigma}{q}$ ratio of 1 when $q = 64$, the noise's standard deviation must also reach 64. More pixels thus become saturated when noise is applied, thus lowering the noise level before even compression. When comparing the output noise to the post-clipping, pre-compression noise, we fall back to a curve similar to the $q = 8$ case.

Although the two curves are similar, a given noise level will have a lower $\frac{\sigma}{q}$ ratio if q is high, and the noise will be more affected. Since high-frequencies components are often attributed higher quantization factors, their noise is thus reduced more.

On the other hand, we observe that textured images, such as traffic, show larger deviations from the theoretical model than non-textured images or even pure synthetic noise. This phenomenon can be explained by the fact that in textured images high frequencies are not only affected by noise but also by texture. Even though our method removes most of the quantization noise, some remain in the residual, and is more prominent when the noise level is comparatively lower.

2.4.4 Discussion

In this section, we derived a model for the effect of JPEG compression on prior noise. Both the theoretical and experimental results show that post-quantization, prior noise is frequency-dependent. In particular, previously-normal noise only remains normal separately for each DCT coefficient. We believe the applications for our study to be numerous and varied, in particular whenever precise knowledge of the noise of an image is required, such as restoration of bad quality (compressed) images, steganography, and image forensics. Here we focused on the luminance channel. Future work will extend this model to the chroma components, which are subsampled in addition to the quantization which was covered here.

2.5 Tracing the Camera Processing Pipeline for Forgery Detection

When analysing the authenticity of an image, one may encounter the traces left by several of the stages presented in Section 2.2 and illustrated in Figure 2.1. Certain approaches rely on metadata to analyse such traces [114]. However, it is important to note that this metadata is frequently altered or eliminated when images are shared online [80]. It is with this consideration in mind that, in the subsequent discussion, we explore algorithms that focus on the pixel-level representation of images, without relying on metadata.

At each stage of the camera pipeline, as depicted in Figure 2.5, distinct traces are left on the image. Tampering within an image results in distinct traces in the authentic and manipulated regions. The detection of these traces depends on the history of the forged image, as various kinds of traces may be impacted and identified. For instance, both demosaicing and JPEG compression introduce periodic artifacts into an image. When an image is forged, local shifts in the phase of these artifacts can occur. However, the detection of demosaicing disruptions [10, 35, 192] is limited to high-quality images that have not undergone significant compression or resampling [10, 181], whereas inconsistencies in JPEG compression [4, 20, 100, 143, 176] are naturally absent in uncompressed images but are more readily discernible in lower-quality versions, such as those shared on social media. Consequently, it becomes feasible to identify alterations in patterns or regions where the compression quality diverges from the rest of the image. These methods are effective when the forgery occurs after an initial compression of the entire image or the inclusion of the forged area. This is often the case with many manipulated images found online, where JPEG images are downloaded, modified, and then re-uploaded, resulting in images that may have undergone multiple compression stages.

Noise analysis can also provide important clues to detect and locate forgeries. As shown in Figure 2.5, each step of the pipeline leaves traces in the noise model. If part of an image has been modified, or comes from a different donor image, the authentic and forged regions are thus likely to present different noise profiles. Figure 2.10 depicts this situation: the forged region presents a different noise model than that of the background image. Several methods aim at detecting this kind of inconsistencies. For instance, Mahdian and Saic [155] perform local wavelet-based noise level estimation using a median absolute deviation estimator.The method developped by Lyu, Pan, and Zhang [154] relies on the kurtosis concentration phenomenon. Still, other noise statistics rather than the noise level can be used for forgery detection. For instance, Itier *et al.*rely on the color noise correlation to detect the boundaries of spliced regions. Splicebuster [49] computes the noise residual of an image after a high-pass filter, and uses the co-occurences of said residuals as local features characterizing the signature of an image. Noiseprint [48] extends on Splicebuster by using Siamese networks to extract a noise-sensitive fingerprint. The extraction of such a fingerprint is further refined in TruFor [90] and further combined with anomaly detection techniques to detect forgeries.

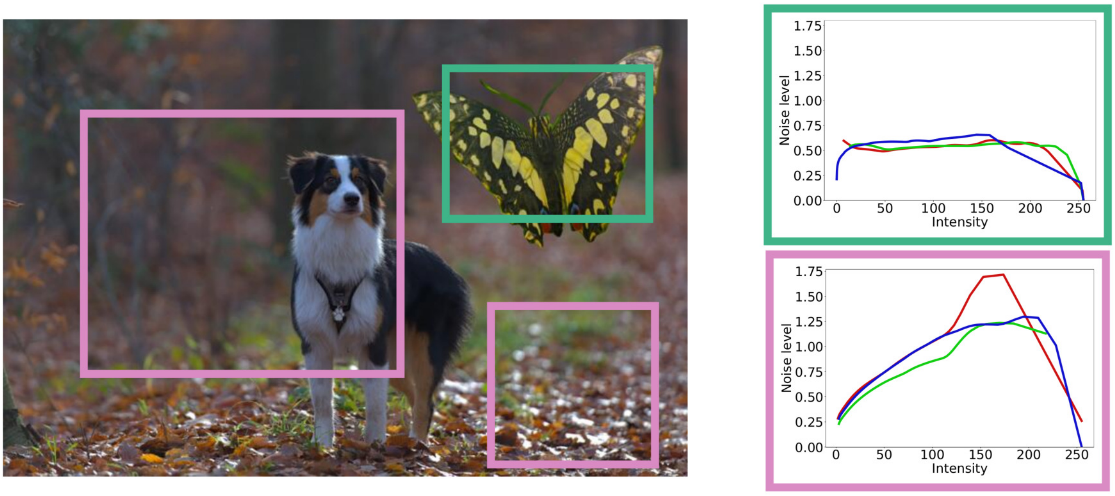


Figure 2.10: Example of a forged image (left) and local noise curves (right). The forged area comes from a different image that has its own pipeline. Noise models (right) differ between the background image (pink) and the donor one (green). The resulting tampered image presents local inconsistencies in the noise model.

The variety of traces that can be present in images make exhaustiveness difficult. Recent works, such as Comprint [158] propose to fuse the characteristic fingerprints of several traces. Another possibility is to consider forgery detection as a learning problem and develop a generic model – usually a neural network – to localize forgeries in the image [99, 162, 221]. While these methods are more generic and potentially more exhaustive, their results are usually opaque. It is thus difficult to know when, and to what extent, they can be trusted. One can also attempt to detect forgeries directly; for instance ManTraNet [221] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localize forgeries in the image. Self- consistency [99] analysis also uses a siamese network with the goal of detecting whether two patches are likely to share the same Exif metadata, and thus to have been processed with the same pipeline. Mayer and Stamm [162] train a siamese network to provide a forensic similarity score. Such score is then used to construct a similarity graph where forgeries are spotted as graph communities (Chapter 5).

Part I

Forgery detection based on the alterations on noise

Chapter 3

Forgery Detection in Digital Images by Multi-Scale Noise Estimation

In this chapter we introduce a simple, yet effective, forgery detection method derived from the observations made in Chapter 2. Indeed, as shown in Section 2.5, tampering within an image is likely to produce inconsistencies in the noise model. The method presented in this chapter looks for such inconsistencies by estimating local noise curves and then comparing them to the global noise curve, obtained from the whole image. This work is published as *Forgery Detection in Digital Images by Multi-Scale Noise Estimation* on J. Imaging [73].

3.1 Introduction

As mentioned in Section 1.1, an escalating number of falsified images are being shared on the web and feeding fake news. Indeed, the popularization of digital devices as well as the development of user-friendly manipulation software have resulted in an increase in the traffic of manipulated content. The credibility of images is under question, and therefore, methods relying on scientific evidence are required to assess the authenticity of images.

Two different approaches have emerged to address this issue. On the one hand, techniques such as digital image watermarking or perceptual image hashing. Image watermarking consists in embedding data into the image that can be later detected or extracted for authentication purposes [16, 195]. On the other hand, perceptual image hashing is a technique used to establish the "perceptual equality" of image content, making it possible to perform image authentication by comparing the hash values of an original image and an image to be authenticated [91, 92].

Although these methods provide reliable authentication, they require an active role from the image owners.

On the other hand, passive methods that do not depend on prior actions or knowledge have also been developed. Starting from the seminal work of H. Farid [64], digital forensics tools have been developed to provide scientific evidence to help determine the authenticity of the images under question [112]. These methods rely on the fact that image forgery techniques leave specific traces (see Chapter 2) that can be detected as local inconsistencies in the image statistics [63, 180]. Most classic methods aim to detect specific cues such as misalignment of the Bayer pattern or perturbations in the demosaicing traces [10, 35, 192], differences in the camera response function [98, 142], or inconsistencies in the JPEG-compression grid or quality [20, 100, 124, 176, 225].

Recent deep-learning models have been developed to tackle the task of forgery detection [25]. These methods can be trained to detect specific falsification techniques such as splicing [19, 188], copy-move [148, 189] and inpainting [137, 213], or to detect general attacks [99, 104, 221]. The main challenge shared by these methods is the construction of adequate training datasets ensuring good results on new real-world examples.

As reviewed in Section 2.5, noise residuals can provide substantial cues for detecting forgeries. Indeed, the initial Poisson noise [67] is transformed by multiple operations specific to each image formation process [40], leading to the final JPEG image. Hence, detecting noise inconsistencies is a rich source of forgery evidence. The use of noise residuals has evolved over time. Early methods [155, 177] directly search inconsistencies in this residual whereas more recent algorithms use it as an input for further feature extraction [48, 49]. Accurately estimating the residual noise traces after the complex set of transformations of the camera's processing chain is the main challenge of this class of algorithms.

With these considerations in mind, we propose a noise-based method built on non-parametric multi-scale noise estimation [41]. The multi-scale approach has been shown to effectively deal with the correlations introduced by the demosaicing and JPEG-compression processes [42] and stands out as a suitable framework for noise inconsistency analysis.

The rest of the chapter is organized as follows. Section 3.2 reviews the image forgery detection techniques based on noise inspection. The proposed method is described in Section 3.3. Section 3.4 presents experimental results in addition to a comparison with other state-of-the-art techniques. The main conclusions are summarized in Section 3.5, where future work directions are also highlighted.

3.2 Related Work

Blind noise-based detection methods usually estimate noise variance locally to detect suspicious regions and then apply a classification criterion to locate forgeries. Mahdian and Saic [155] propose to estimate the noise variance in blocks using a median absolute deviation (MAD) estimator in the wavelet domain. Classification is performed using homogeneous noise standard deviation criteria. In turn, Ke *et al.* [113] propose noise level estimation using principal component analysis (PCA) [185]. K-means is then applied to group image blocks into two clusters. A similar approach is developped by Zeng *et al.* [232]. A different method was introduced by Lyu *et al.* [154], where block-wise noise estimation is based on the observation that the kurtosis values across different band-passed filter channels are constant [242]. The method concludes by segmenting the image into regions with significantly different noise variances by k-means. Liu *et al.* [144], segment the input image using the simple linear iterative clustering (SLIC) algorithm. Then, for each region, five filters are used to extract noise. The computed noise features are then used for classification, which is performed by energy-based graph cut.

The aforementioned methods estimate a single and constant noise level, namely an additive white Gaussian noise (AWGN) model. However, this hypothesis does not hold in realistic scenarios since noise levels depend on the image intensity (see Section 2.3). More recent methods consider this fact and estimate a noise level function (NLF) rather than a single noise level. Yao *et al.* [224] proposed to jointly estimate the NLF and the camera response function (CRF) by segmenting the image into edge and non-edge regions. Noise level functions are then compared and an empirical threshold is fixed in order to detect salient curves. The methods introduced by Julliand *et al.* [106, 108] instead analyze a histogram based on the noise density function at the local level in order to reveal suspicious areas. The method proposed by Pun *et al.* [184] computes an NLF-based on Wiener filtering. Local noise levels in regions with a certain brightness are assumed to follow a Poisson distribution, according to which, the larger the distance to the NLF, the higher the probability of forgery. On the other hand, the approach developed by Zhu and Li [241] consists of estimating a noise level function that depends on the local sharpness rather than on the intensity.

Recently, forgery detection methods based on deep learning and feature modeling have been developed. Cozzolino *et al.* [49] propose using noise residuals to extract local features and compute their co-occurrence histograms, which are then classified in two classes using the expectation–maximization algorithm. More recently, the same authors presented a novel CNN-based method for noise residual extraction [48]. A similar approach can be found in [164], which will be further analysed in Chapter 5. On the other hand, Zhou *et al.* [240] proposed a two-stream CNN, one for the detection of tampering artifacts and the other to leverage noise features. Deep learning-based methods are more general than previously described ones. A major limitation of these methods is that they require large training datasets, which are not always available. Furthermore, their performance generally remains dataset dependent.

3.3 The Proposed Method

We propose a new method for JPEG-compressed image forgery detection based on multi-scale noise estimation. The method addresses the fact that, after going through the complete camera processing pipeline, noise is not only signal-dependent but also frequency-dependent (see Sections 2.3 and 2.4). In particular, after demosaicing, noise becomes spatially correlated, and furthermore, the quantization of the DCT coefficients during JPEG-compression differently affects the noise at each frequency. In this context, multi-scale noise estimation is the most suitable approach since it enables capturing noise at medium and low frequencies.

Let I be an image with C color channels. We first split the image into $W \times W$ blocks with $1/2$ overlap, extending the image in the borders by mirroring if necessary. We will refer to these blocks as macroblocks. For each color channel, we estimate the global image noise curve as well as the local noise curves for each macroblock using the extension of the Ponomarenko *et al.*method [41]. For each channel, we compare the global noise curve with the ones locally obtained by computing the number of bins of the local noise curve that are below the global noise curve. By doing so, we obtained a heatmap for each channel that shows, for each macroblock, the percentage of bins in its noise curve whose count is below the global estimation. The information contained in the C obtained heatmaps is then combined by taking their geometric mean. As a result, we obtain a single heatmap.

For non-forged images, we expect the macroblocks to show similar noise levels functions as the one computed for the whole image. However, noise estimation is highly affected by image content. Indeed, noise overestimation is expected to happen in textured regions [146]. As a consequence, local noise curves computed over textured areas may be above the global one, even if no tampering has been performed. To prevent this kind of macroblock being perceived as suspicious, we only consider the number of bins below the global noise curve. Indeed, the global noise curve provides a lower bound for local noise curves since the noise estimation algorithm [41] has more samples from which to choose the adequate ones to estimate noise. Therefore, local noise curves that are below the global one are suspected to correspond to a different source. Figure 3.1 depicts the previously described situation. Indeed, we can observe that the non-forged macroblock shows higher noise levels than the global image, even though it is not tampered. On the other hand, the manipulated macroblock exhibits lower noise levels.

The next step consists of repeating the previously described process but replacing the image I and the macroblocks by their down-scaled version. To this aim, let S be the operator that tessellates the image into sets of 2×2 pixels blocks, and replaces each block by the average of the four pixels. We define $S_n(I)$as the n-th scale of an image I obtained by applying n

times the operator S to the image I. This procedure allows noise curves to show the noise contained in lower frequencies and can provide further evidence of tampering that could be hidden under strong JPEG-compression.

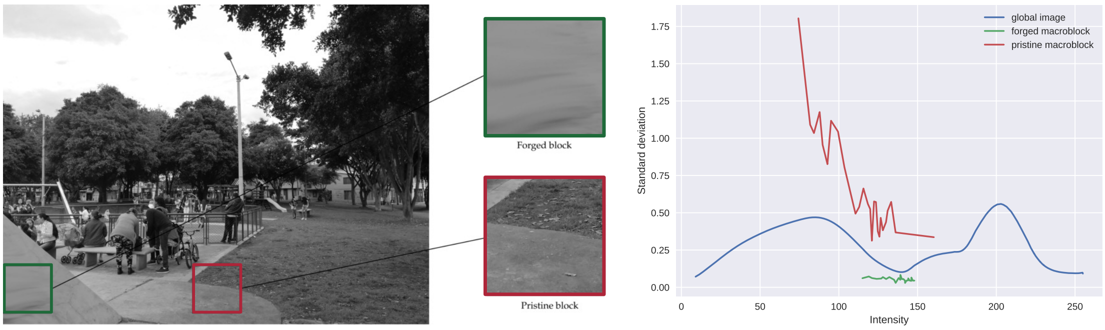


Figure 3.1: Estimated noise curves for the global image and for two macroblocks—one of which is contained in the manipulated region and the other is coming from the non-manipulated part of the image.

By iterating the process at successive scales, we obtain one heatmap per scale which shows the geometric mean of the percentages obtained at each channel. Each of these heatmaps may provide useful information to detect tampering since they account for noise contained at different frequencies. The sum of the heatmaps obtained at the different scales is computed and then normalized in the $[0, 255]$ interval. To obtain the final heatmap, for each pixel we compute the average of the values of each macroblock containing it.

The residual noise present in images having undergone demosaicing and JPEG-compression is correlated and therefore creates medium-sized noise spots. This may cause the blocks of size 8×8 used for noise estimation to fit inside these spots, thus causing noise underestimation. Again, estimating noise in sub-sampled versions of the image enables these spots to fit inside the scanning blocks and to accurately measure low-frequency noise. We propose repeating the sub-scaling process until reaching $S_2(I)$, as suggested in [42].

Further scales could be also considered. However, the most relevant information is already retrieved at S_2. Furthermore, the macroblock's size would become critically small and unfit to estimate noise curves: if the original macroblocks are sized $W \times W$ in S_0, in S_1 they will be of size $(W/2) \times (W/2)$, and in S_2 of size $(W/4) \times (W/4)$. Indeed, as shown in Section 3.4.3, the best performance for the proposed method is achieved when considering macroblocks of size $W = 256$. In this context, the macroblocks are sized 128×128 in S_1 and 64×64 in S_2.

Figure 3.2 shows the pipeline of the proposed method, from the moment that the algorithm is fed with the input image until the final heatmap is delivered. Additionally, a summarized version of the proposed method is given in Algorithm 1. The actual source code is available at `https://github.com/marigardella/PB_Forgery_Detection`, together with the instructions and requirements to run the method.

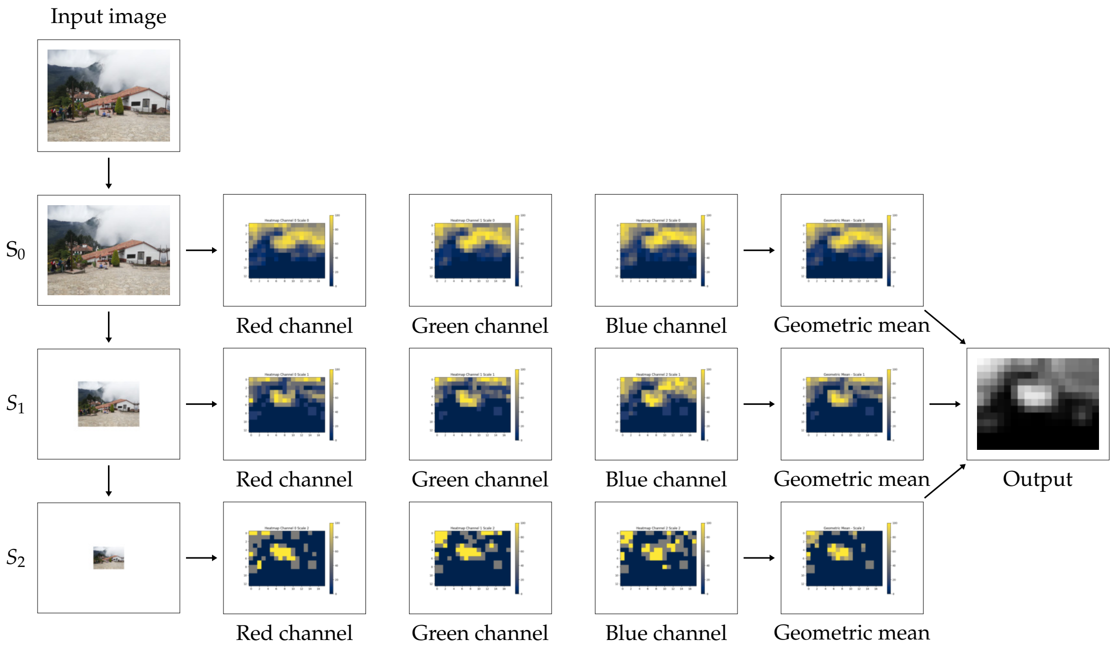


Figure 3.2: Complete pipeline of the method: successive scales are extracted from the input image. At each scale, one heatmap per color channel is computed and then combined according to their geometric mean. Finally, the obtained heatmaps at each scale are summed and normalized to produce the final output.

Algorithm 1: Pseudo-code for the proposed method

```
Input I: image of shape $N_x \times N_y$ with $C$ color channels.
Param W = 256: macroblock side
Param S= 0.5: stride
Param num_scales: number of scales

$M_x = \lfloor N_x/(W \times S) \rfloor - 1.$        # horizontal number of macroblocks
$M_y = \lfloor N_y/(W \times S) \rfloor - 1.$        # vertical number of macroblocks
macroblocks_list ← list of all $W \times W$ macroblocks with $S$ stride.
for each scale $s$ do
    for each channel $c$ do
        $I_s^c \leftarrow$ get image in scale $s$ and channel$c$.
        $f_{I_s^c} \leftarrow$ noise curve estimation for $I_s^c$ using [41].
        $H^c \leftarrow$ zeros$(M_x \times M_y)$.
        for each macroblock in macroblocks_list do
            $M_s^c \leftarrow$ get macroblock in scale $s$ and channel $c$.
            $f_{M_s^c} \leftarrow$ noise curve estimation for $M_s^c$ using [41].
            $H^c[M_s^c] \leftarrow$ percentage of bins of $f_{M_s^c}$ below $f_{I_s^c}$.
    $H_s \leftarrow$ geometric mean of the heatmaps $H^c$.
$H_{\text{aux}} \leftarrow$ sum and normalization of heatmaps $H_s$.
$H \leftarrow$ compute for each pixel the average of $H_{\text{aux}}$ for each macroblock containing it.
return $H$.
```

3.4 Experimental Results

We conducted three experiments. First, we evaluated the relevance of the multi-scale approach by comparing the results obtained using a single scale $(S_0(I))$, two sub-scales $(S_0(I)$ and $S_1(I))$ and three sub-scales $(S_0(I)$, $S_1(I)$ and $S_2(I))$. Second, we compared our method with state-of-the-art forgery-detection algorithms based on noise analysis. Finally, we evaluate the influence of W, the main parameter of the method, in the performance of the proposed approach.

Datasets All experiments were conducted on the CG-1050 database [26] which contains four datasets, each one corresponding to a different forgery technique: colorization, copy-move, splicing and retouching. The total number of forged images is 1050. This database is varied in nature, including images captured in 10 different places. The size of the images is 3456×4608 or 4608×3456 pixels. The database includes both RGB and grayscale images, all of which are JPEG-compressed. The estimated JPEG-quality [124] for each dataset is shown in Table 3.1.

Table 3.1: Average JPEG-quality and range for each of the datasets.

	Retouching	**Colorization**	**Splicing**	**Copy-Move**
Average JPEG-quality	86.9	86.8	87.3	86.8
JPEG-quality range	[71,88]	[71,88]	[71,88]	[71,88]

Forgery masks were constructed by computing the absolute difference between the original image and the forged one in each channel. To avoid pixels whose values had changed due to global manipulations rather than tampering, the difference from one image to another was thresholded. Only pixels whose value varied more than this threshold for at least one channel were kept. Masks were then further refined in order to prevent isolated pixels from being regarded as forged. The thresholds used were 15 for the copy-move, colorization and splicing datasets and 10 for the retouching one.

The distribution of the mask's size on each of the four datasets is shown in Figure 3.3.

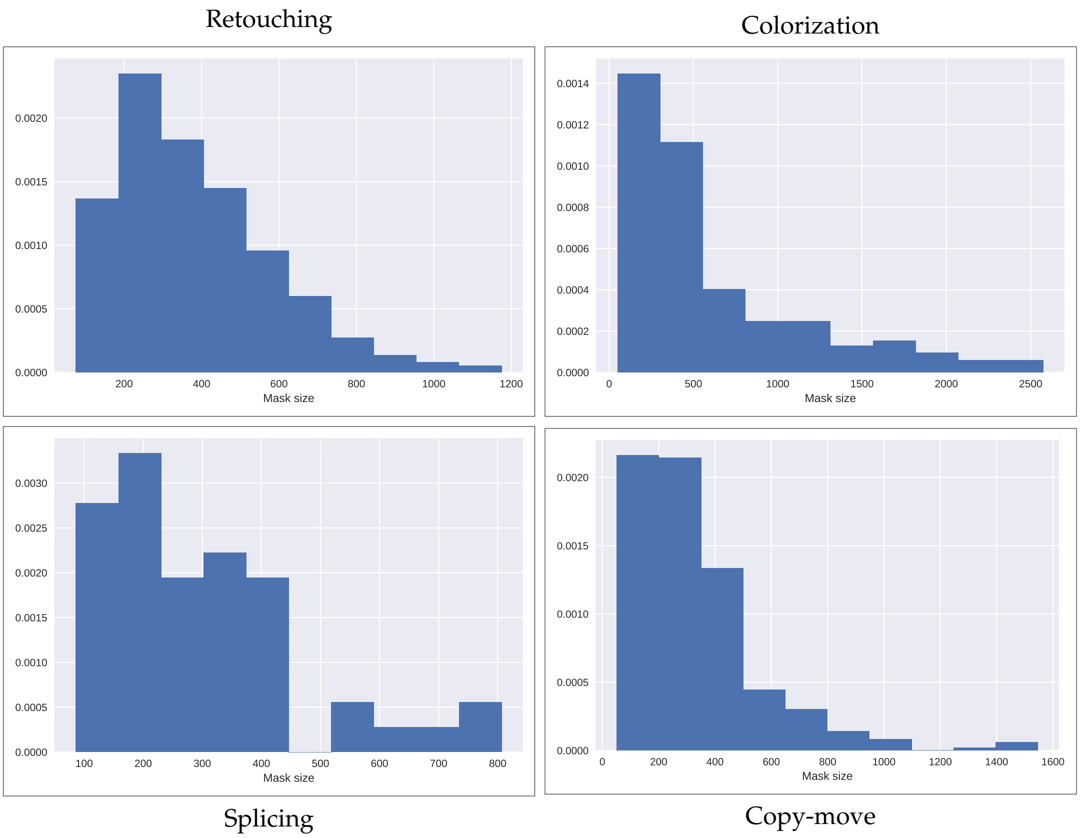


Figure 3.3: Distribution of the forgery size in each of the datasets considered. The forgery size is shown as the square root of the mask size, which represents the side of its equivalent square.

Evaluation Measures Forgery localization is a particular case of binary classification. Indeed, there are two possible classes for each pixel: forged (positive) or non-forged (negative). Performance measures are usually based on the confusion matrix [198], which has four values, each one corresponding to the four possible combinations of predicted and actual classes, as shown in Figure 3.4.

		Predicted class	
		Positive	*Negative*
Actual class	*Positive*	True positive	False negative
	Negative	False positive	True negative

Figure 3.4: Confusion matrix: rows represent the actual classes while columns represent the prediction. The matrix has four possible values, corresponding to the four possible combinations of predicted and actual classes.

Three metrics based on these four quantities are proposed in order to compare the results obtained in both experiments. Namely, we evaluated the results using the IoU, the F1 and the

MCC scores, defined as

$$\begin{aligned}\mathsf{MCC} &= \frac{\mathsf{TP} \times \mathsf{TN} - \mathsf{FP} \times \mathsf{FN}}{\sqrt{(\mathsf{TP} + \mathsf{FP}) \times (\mathsf{TP} + \mathsf{FN}) \times (\mathsf{TN} + \mathsf{FP}) \times (\mathsf{TN} + \mathsf{FN})}}\,,\\ \mathsf{IoU} &= \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}\,,\\ \mathsf{F1} &= \frac{2\mathsf{TP}}{2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}}\,.\end{aligned}$$

where TP stands for true positive, TN for true negative, FN for false negative and FP for false positive.

These metrics are designed to evaluate binary-estimated masks. However, all of the methods analyzed in this chapter propose continuous heatmaps rather than binary masks. To adapt the metrics to the continuous setting, we used their weighted version. In this approach, the value of a heatmap H at each pixel x is regarded as the probability of forgery of the pixel. Therefore, we define the weighted true positives, weighted true negatives, weighted false negatives and weighted false positives as:

$$\begin{aligned}\mathsf{TP}_w &= \sum_x H(x) \times M(x),\\ \mathsf{TN}_w &= \sum_x (1 - H(x)) \times (1 - M(x)),\\ \mathsf{FN}_w &= \sum_x H(x) \times (1 - M(x)),\\ \mathsf{FP}_w &= \sum_x (1 - H(x)) \times M(x),\end{aligned}$$

respectively, where H is the output heatmap normalized between 0 and 1, and M is the ground-truth binary mask where pixels with a value of 1 are forged. Then, the weighted version of the IoU, F1 and MCC scores are obtained replacing TP, TN, FN and FP with their weighted versions. It is important to point out that for some of the methods, the output is a two-sided heatmap (meaning that suspicious regions can appear in lighter or darker colors). Taking this into consideration, both the output heatmap and the inverted one are evaluated and only the highest score is kept.

3.4.1 Relevance of the Multi-Scale Approach

We first examined the pertinence of a multi-scale scheme. For this purpose, we computed the results obtained when considering one single scale $S_0(I)$ (which would correspond to the input image), using two scales $S_0(I)$ and $S_1(I)$, and using three scales $S_0(I), S_1(I)$ and $S_2(I)$. The scores obtained for each of these settings are shown in Table 3.2.

We can observe that using multiple scales leads to better results compared to a single one. Indeed, in all four datasets, the scores obtained by PB2 and PB3 are better than those obtained by PB1 for the three metrics. Regarding the number of scales yielding a better performance, the use of three scales obtains the best scores for the retouching, colorization and splicing datasets, whereas the use of two scales achieves a better performance in the copy-move dataset. However, the results obtained for the copy-move dataset are poor for the three variants of the method, and furthermore, they have very similar scores. We conclude that the use of three scales, $S_0(I)$, $S_1(I)$ and $S_2(I)$, gives the best performance among the evaluated alternatives. In fact, given that JPEG-compression is applied in 8×8 blocks without overlap, it is at S_2 that the most accurate noise estimation is achieved since we are able to capture noise contained in lower frequencies, which is less affected by the quantization of the DCT coefficients.

		MCC		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1	0.0672	0.0958	0.0276	**0.0380**
PB2	0.0848	0.1066	0.0310	0.0377
PB3	**0.0915**	**0.1108**	**0.0316**	0.0362
		IoU		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1	0.0242	0.0721	0.0112	0.0148
PB2	0.0284	0.0756	0.0122	**0.0149**
PB3	**0.0300**	**0.0761**	**0.0123**	0.0145
		F1		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1	0.0454	0.1122	0.0216	0.0281
PB2	0.0529	0.1175	0.0234	**0.0282**
PB3	**0.0557**	**0.1192**	**0.0236**	0.0276

Table 3.2: MCC, IoU and F1 scores for our method with one scale (PB1), two scales (PB2) and three scales (PB3).

3.4.2 Comparison with State-of-the-Art Methods

In order to assess the performance of our method, we compared the results obtained on the CG-1050 dataset with those delivered by state-of-the-art noise-based methods: Splicebuster [49], Noiseprint [48], Mahdian [155], Pan [177], Zeng [232], Zhu [241] and Median [210]. For each algorithm, we used a publicly available implementation [230]. Table 3.3 lists all the evaluated methods as well as their reference article and the link to the source code used for the comparison.

Method	**Ref.**	**Source Code**
Mahdian	[155]	`https://github.com/MKLab-ITI/image-forensics`
Pan	[177]	`https://github.com/MKLab-ITI/image-forensics`
Zeng	[232]	`https://github.com/MKLab-ITI/image-forensics`
Median	[210]	`https://github.com/MKLab-ITI/image-forensics`
Splicebuster	[49]	`http://www.grip.unina.it/research/83-multimedia_forensics`
Noiseprint	[48]	`http://www.grip.unina.it/research/83-multimedia_forensics`
Zhu	[241]	`https://github.com/marigardella/Zhu_2018`

Table 3.3: State-of-the-art methods used for the comparison as well as their reference and link to source code.

The obtained results are given in Table 3.4. We observe that Splicebuster outperforms the rest of the methods in the retouching and splicing datasets regardless of the metric.

			MCC		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**	**Average Ranking**
PB3	0.0915 (2)	**0.1108** (1)	0.0316 (2)	**0.0362** (1)	1.5
Splicebuster	**0.1176** (1)	0.0535 (4)	**0.0502** (1)	0.0233 (4)	2.5
Mahdian	0.0434 (6)	0.0566 (3)	0.0247 (4)	0.0257(3)	4
Pan	0.0513 (4)	0.0681 (2)	0.0282 (3)	0.0306 (2)	2.75
Noiseprint	0.0558 (3)	0.0361 (6)	0.0182 (6)	0.0177 (6)	5.25
Median	0.0479 (5)	0.0469 (5)	0.0204 (5)	0.0195 (5)	5
Zeng	0.0180 (7)	0.0262 (7)	0.0119 (8)	0.0117 (8)	7.5
Zhu	0.0147 (8)	0.0201 (8)	0.0180 (7)	0.0123 (7)	7.5
			IoU		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**	**Average Ranking**
PB3	0.0300 (3)	**0.0761** (1)	0.0123 (2)	0.0145 (2)	2
Splicebuster	**0.0600** (1)	0.0577 (2)	**0.0242** (1)	**0.0166** (1)	1.25
Mahdian	0.0168 (5)	0.0548 (4)	0.0102 (5)	0.0131(5)	4.75
Pan	0.0198 (4)	0.0576 (3)	0.0109 (4)	0.0138 (4)	3.75
Noiseprint	0.0312 (2)	0.0450 (7)	0.0114 (3)	0.0142 (2)	3.5
Median	0.0163 (6)	0.0513 (5)	0.0095 (7)	0.0123(6)	6
Zeng	0.0136 (7)	0.0441 (8)	0.0084 (8)	0.0114 (8)	7.75
Zhu	0.0129 (8)	0.0453 (6)	0.0102 (5)	0.0116(7)	6.5
			F1		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**	**Average Ranking**
PB3	0.0557 (3)	**0.1192** (1)	0.0236 (2)	0.0276 (2)	2
Splicebuster	**0.1081** (1)	0.0965 (2)	**0.0448** (1)	**0.0314** (1)	1.25
Mahdian	0.0324 (5)	0.0902 (4)	0.0199 (6)	0.0250(5)	5
Pan	0.0380 (4)	0.0946 (3)	0.0211 (4)	0.0264 (4)	3.75
Noiseprint	0.0588 (2)	0.0778 (7)	0.0222 (3)	0.0271 (3)	3.75
Median	0.0315 (6)	0.0857 (5)	0.0185 (7)	0.0236 (6)	6
Zeng	0.0264 (7)	0.0765 (8)	0.0165 (8)	0.0220 (8)	7.75
Zhu	0.0250 (8)	0.0779 (6)	0.0200 (5)	0.0224(7)	6.5

Table 3.4: Results of the evaluated methods measured by the average weighted IoU, F1 and MCC scores for each dataset that maximized the score.

Our method ranks first for colorization attacks for all the three metrics considered. This forgery technique shows the relevance of considering noise curves instead of single noise levels. Indeed, when changing the color in a region of the image, noise levels are not necessarily perturbed. However, those noise levels will not be consistent with the new intensity but with the original. Estimating noise curves as the proposed method does enables detecting this kind of inconsistency which only appears when considering intensity-dependent noise models. Regarding the copy-move dataset, Splicebuster delivers the best results when considering the F1 and IoU scores. However, our approach obtains the best MCC score.

The average ranking shows that Splicebuster outperforms the rest of the methods when considering both the F1 and IoU scores, followed by our method. Nevertheless, our method achieves the best average ranking when considering the MCC score, followed by Splicebuster. Noiseprint stands out as the third best performing method for the IoU and F1 scores. It even ranks second for retouching and copy-move attacks when considering these scores. However, it shows a poor performance for the colorization dataset. This can be explained by the fact that the camera signature is left unchanged when performing this kind of manipulation. The Pan and Mahdian methods are middle-ranked, showing better results when considering the MCC score. Finally, Median, Zeng and Zhu show the worst performance of all the considered methods regardless of the metric considered.

All of the evaluated methods have different resolutions which may affect their performance when forgeries are too small. To analyze the effects of the size of the forgeries, we computed the average score as a function of the forgery size. Figure 3.5 shows the average score obtained by each method when setting different lower bounds for the forgery size in each of the datasets considered.

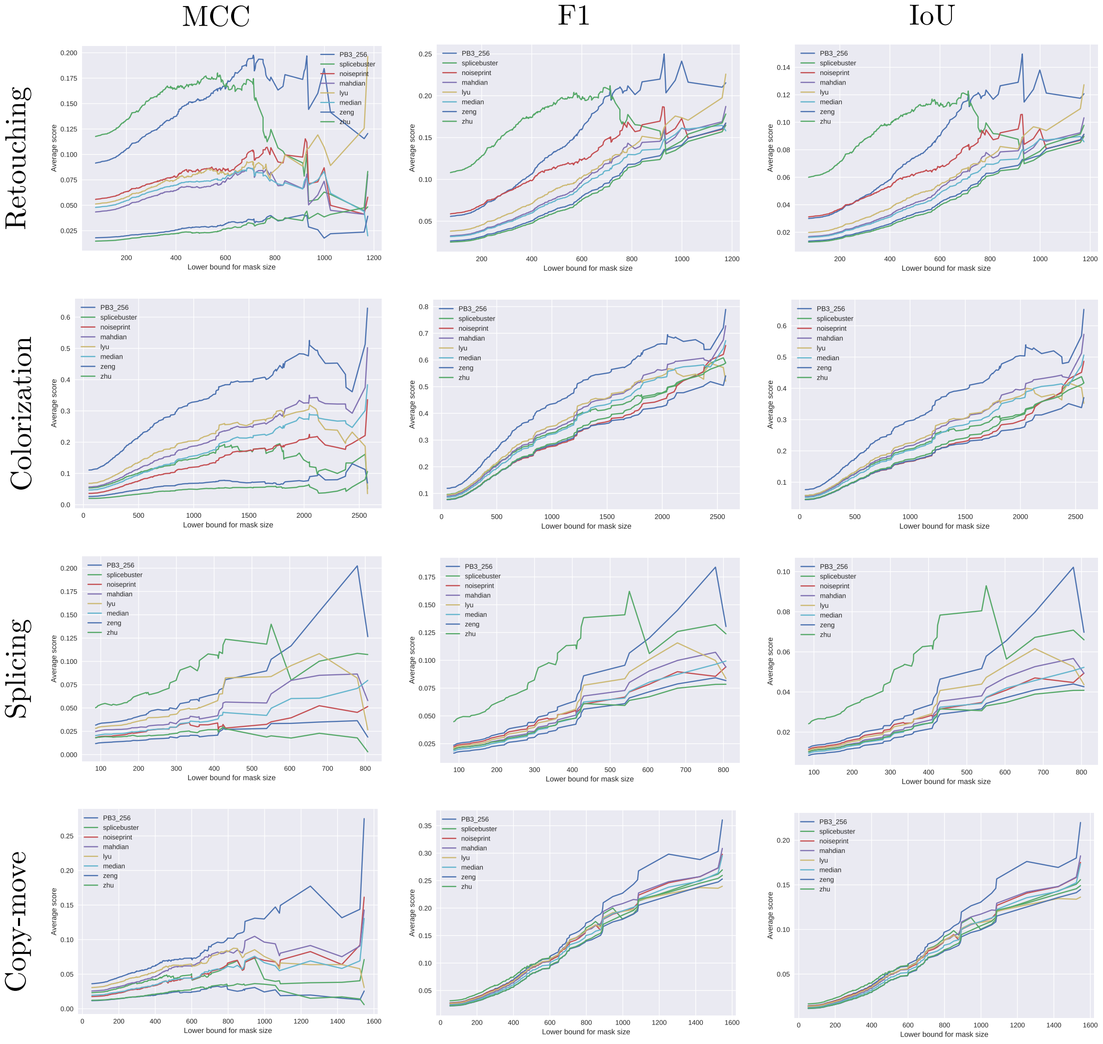


Figure 3.5: Average weighted MCC (left), IoU (middle) and F1 (right) scores obtained by each method as a function of the lower bound for the forgery size, in each of the datasets considered. Forgery size is shown as the square root of the mask size, which represents the side of its equivalent square.

The results suggest that our method outperforms the state-of-the-art approaches when considering large forgeries in all the datasets regardless of the considered score. The fact that it does not perform that well when considering small manipulations is a direct consequence of the size of the macroblocks. Indeed, for our method to provide reliable detection, the tampered region should be at least of the size of one of the tested macroblocks. In contrast, the performance of Splicebuster decreases as we consider larger forgeries. This is partially expected since the Gaussian-uniform model used in this method is better suited for small forgeries, as suggested by their authors in the original paper [49].

For further evaluation, we used the visual inspection of the results obtained by the proposed method and state-of-the-art approaches. Figure 3.6 shows examples of the outputs obtained by these methods for the colorization and retouching attacks, respectively, as well as for the corresponding original untampered images.

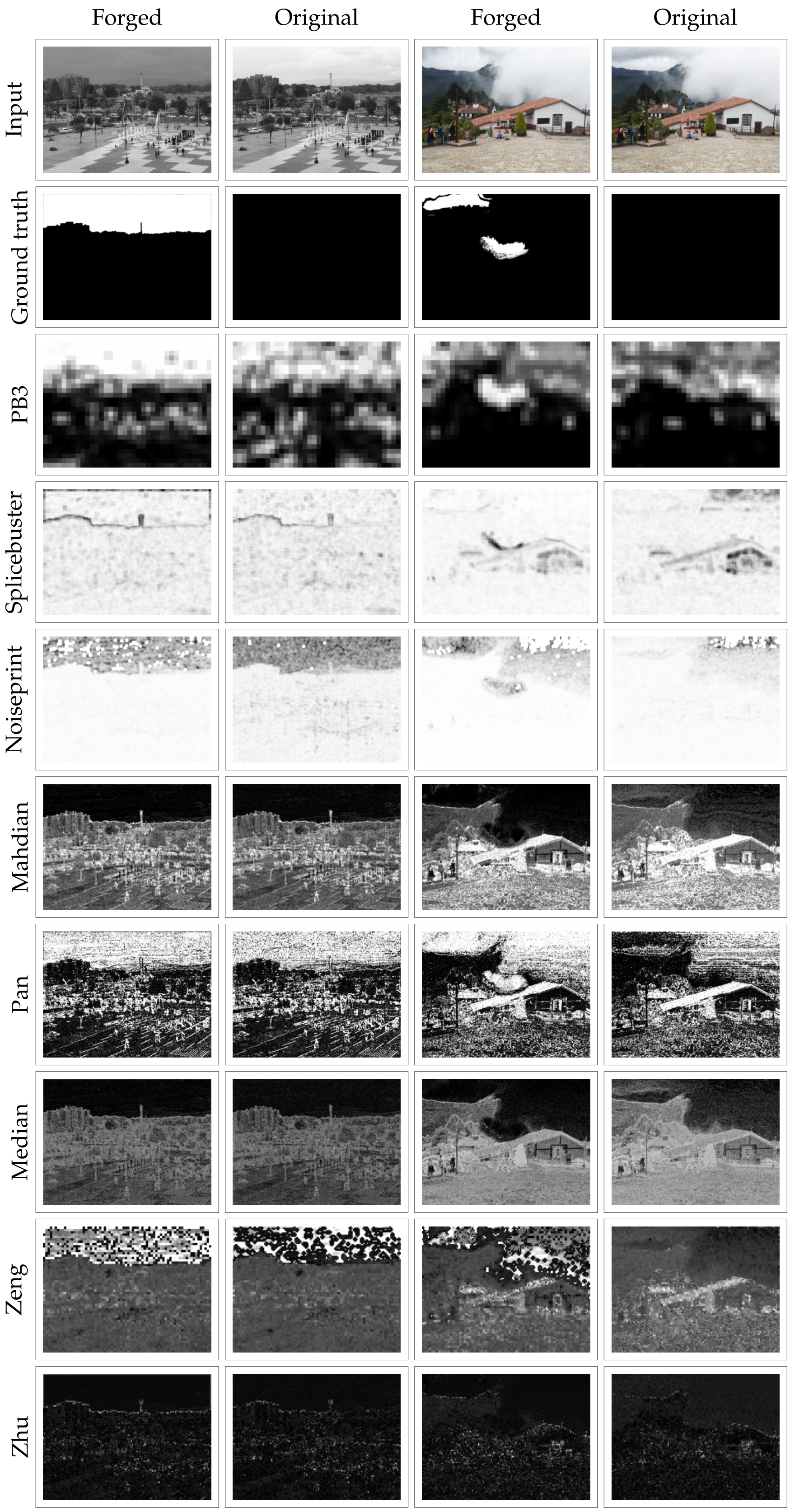


Figure 3.6: Results obtained for examples where colorization (first column) and retouching (third column) were performed, as well as for their corresponding original images (second and fourth columns). On the successive rows, the results obtained by each of the approaches for these images.

For the colorization attack shown in Figure 3.6, we can observe that, for all of the approaches except ours, the heatmap obtained when applying the method to the forged and original images are very similar. None of these methods is able to distinguish the tampered region by detecting the traces of the forgery. Instead, the proposed method provides a significant difference between the forged and pristine image; we observe that the forgery clearly stands out while for the pristine image, the values of the heatmap in that area are moderated.

In the case of retouching, we observe that all of the methods point out the forged region or at least part of it as suspicious. However, several interpretation problems arise. When analyzing the results provided by Splicebuster, we can notice that the heatmap corresponding to the tampered image precisely points to the border of part of the forgery. However, when considering the pristine image, there are several areas of the heatmap showing the same values, even if they are not tampered. The Noiseprint results better localize the forgery even though false alarms are present in the pristine image. Mahdian, Pan, Median, Zeng and Zhu methods show a further drawback: in the heatmap corresponding to the manipulated image, the forged regions stand out at the same level as other non-tampered parts of the image. The interpretation of the heatmaps is left to the user who has to decide whether the regions detected as suspicious should be considered forged or discarded. On the other hand, our method is able to localize the forgery when applied to the tampered image while showing no extreme values for the pristine one, making it easier for users to interpret.

3.4.3 Influence of the Macroblock Size

The main parameter of the proposed method is W, the size of the macroblocks where local noise curves are computed. The larger this size, the more accurate the NLF estimation. However, the size of the macroblocks directly affects the precision with which forgeries are located. As shown in Figure 3.5, the performance of the method relies on the macroblocks' size.

In order to evaluate the capabilities of the method, we carried out an analysis of such performance depending on the size of the macroblocks. We tested three possible values for W: 512, 384 and 256. The results, presented in Table 3.5, suggest that the best performance is achieved for $W = 256$. Indeed, for the retouching, colorization and copy-move datasets, the best scores are obtained when considering macroblocks of size 256×256. On the other hand, when considering the splicing dataset, macroblocks of size 512×512 yield a better IoU score. However, the difference is very small and when considering other metrics, $W = 256$ achieves higher scores.

		MCC		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1_512	0.0585	0.0770	0.0246	0.0316
PB2_512	0.0729	0.0830	0.0268	0.0321
PB3_512	0.0804	0.0901	0.0291	0.0320
PB1_384	0.0625	0.0838	0.0242	0.0348
PB2_384	0.0789	0.0924	0.0284	0.0350
PB3_384	0.0869	0.1015	0.0289	0.0344
PB1_256	0.0672	0.0958	0.0276	**0.0380**
PB2_256	0.0848	0.1066	0.0310	0.0377
PB3_256	**0.0915**	**0.1108**	**0.0316**	0.0362
		IoU		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1_512	0.0226	0.0650	0.0113	0.0141
PB2_512	0.0262	0.0673	0.0120	0.0144
PB3_512	0.0278	0.0691	**0.0124**	0.0142
PB1_384	0.0234	0.0679	0.0110	0.0145
PB2_384	0.0274	0.0708	0.0120	0.0146
PB3_384	0.0289	0.0730	0.0122	0.0144
PB1_256	0.0242	0.0721	0.0112	0.0148
PB2_256	0.0284	0.0756	0.0122	**0.0149**
PB3_256	**0.0300**	**0.0761**	0.0123	0.0145
		F1		
	Retouching	**Colorization**	**Splicing**	**Copy-Move**
PB1_512	0.0428	0.1032	0.0215	0.0268
PB2_512	0.0492	0.1067	0.0229	0.0272
PB3_512	0.0520	0.1099	0.0235	0.0270
PB1_384	0.0441	0.1068	0.0211	0.0275
PB2_384	0.0512	0.1112	0.0229	0.0277
PB3_384	0.0540	0.1151	0.0232	0.0274
PB1_256	0.0454	0.1122	0.0216	0.0281
PB2_256	0.0529	0.1175	0.0234	**0.0282**
PB3_256	**0.0557**	**0.1192**	**0.0236**	0.0276

Table 3.5: MCC, IoU and F1 and scores for our method with one scale (PB1), two scales (PB2) and three scales (PB3) and considering different macroblock sizes: 512, 384 and 256.

3.5 Conclusions and Limitations

In the fight against disinformation, the use of objective methods able to detect manipulated multimedia content becomes crucial. Providing such tools is the aim of the digital forensics research community, and in particular, of the present work. We believe that image forgery detection is a key resource to fight fake news.

JPEG images are broadly used and clearly stand out as one of the most popular image formats. From the acquired raw image to the final JPEG format delivered by the camera, a complex processing chain is applied. Along this process, the originally Poisson-distributed noise undergoes several transformations, resulting in a complex noise structure in the JPEG image whose model does not match the AWGN hypothesis. Noise inconsistency analysis is a rich resource for forgery detection given that forged regions are likely to have undergone a different processing pipeline or an out-of-camera manipulation. However, noise-based methods require accurately dealing with the changes induced by the successive steps of the camera processing chain.

In this chapter, we presented a method that can correctly deal with the complex noise residuals observable in the JPEG image. The proposed method implements a multi-scale approach which has shown to be suitable for analyzing the highly correlated noise present in JPEG-compressed images.

Our comparative results show that our method outperforms state-of-the-art approaches when evaluating the results with the MCC score. For colorization attacks, our method performs best, regardless of the metric. In addition, when the size of the forgeries is large enough, our method shows the best performance in all the datasets, for all three considered metrics.

Nevertheless, the proposed method has its own limitations, mainly related to too-small and too-large forgeries. Indeed, if the forgery is too small with respect to the macroblock's size, the method is likely to miss it. On the other hand, if the forgery is comparatively too large, the global noise curve may be distorted by the tampered region. The method is also by construction unable to detect a pure internal copy-move. Indeed, such a manipulation leaves the noise model unaltered. As a final negative note, the method cannot detect splicing when the forged region has more noise than the background image.

Chapter 4

Noisesniffer: Forgery Detection by Noise Spatial Statistics

In this chapter we present a more sophisticated forgery detection method based on noise analysis. This method estimates for each image a background stochastic model which makes it possible to detect local noise anomalies. The algorithm includes an a contrario statistical validation step, which associates a Number of False Alarms (NFA) with each tampering detection. Detections are obtained by a threshold of the NFA, which renders the method fully automatic and endows it with a false alarm control mechanism.

This work is accepted for publication as *Image forgery detection based on noise inspection: analysis and refinement of the Noisesniffer method* on IPOL which is an improvement over our work published in the IWBF conference [72]. An online demo is available at: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000341

4.1 Introduction

As reviewed in Chapter 2, the residual noise present in images depends on the in-camera processing chain. It can therefore reveal the presence of forged regions by detecting local inconsistencies in the noise statistics that are incompatible with a unique camera processing pipeline. Such inconsistencies can be produced by the forgery or its post-processing. In the previous chapter we presented a simple yet effective method to exploit noise inconsistencies for forgery detection. Here, we present a more sophisticated forgery detection method based on noise analysis. Hereafter, we shall focus on the relevant references for the particular topic of this chapter. To make the discussion self-contained, we opted to keep some references that might have already been presented in the previous chapter.

Blind noise-based detection methods usually estimate noise variance locally to detect suspicious regions and then apply a classification criterion to locate forgeries. Mahdian and Saic [155] propose a block-wise noise variance estimation using a median absolute deviation (MAD) estimator in the wavelet domain. Their classification process relies on a homogenous noise standard deviation criterion. In contrast, Ke et al. [113] estimate the noise level using principal component analysis (PCA) [185]. They employ K-means clustering to group image blocks into two clusters. A similar approach is developed by Zeng et al. [232]. A different method was introduced by Lyu et al. [154], where block-wise noise estimation is based on the observation that the kurtosis values across different band-passed filter channels are constant [242]. The method concludes by segmenting the image into regions with significantly different noise variances by k-means. Liu and Pun [144] employ a different segmentation technique known as the simple linear iterative clustering (SLIC). For each region, they use five filters to extract noise characteristics, and subsequently, the computed noise features are used for classification via an energy-based graph cut method.

The aforementioned methods share the same drawback: they estimate a single and constant noise level, namely an additive white Gaussian noise (AWGN) model. Yet this hypothesis does not hold in realistic scenarios where noise levels depend on the image intensity [147]. More recent methods consider this fact and estimate a noise level function (NLF) rather than a single noise level. Yao et al. [224], proposed to jointly estimate the NLF and the camera response function (CRF) by segmenting the image into edge and non-edge regions. Noise level functions are then compared and an empirical threshold is fixed to detect salient curves. The methods introduced by Julliand et al. [106, 108] instead analyze a histogram based on the noise density function at the local level in order to reveal suspicious areas. Pun et al. [184] compute an NLF-based on Wiener filtering. Local noise levels in regions with a certain brightness are assumed to follow a Poisson distribution, according to which, the larger the distance to the NLF, the higher the probability of forgery. On the other hand, the approach introduced by Zhu and Zhao [241] consists of estimating a noise level function that depends on the local sharpness rather than on the intensity. Gardella et al. [73] suggest that, for JPEG-compressed images, not only the noise is intensity dependent but also scale-dependent. Thus, they propose to estimate an NLF at different scales to capture noise inconsistencies at multiple scales.

Recently, forgery detection methods based on deep learning and feature modeling have been developed. Splicebuster [49] proposes to use noise residuals to extract local features and compute their co-occurrence histograms, which are then classified in two classes using the Expectation–Maximization algorithm. More recently, the same authors presented a novel CNN-based method for noise residual extraction [48]. A similar approach was introduced by Mayer and Stamm [70, 164], where a pair of CNNs in a siamese configuration is used to extract camera-related artifacts from pairs of patches which are then used to construct a similarity graph. Forgeries are detected as communities in the aforementioned graph. Zhou et al [240] proposed a two-stream CNN, one for the detection of tampering artifacts and the other to leverage noise features. Deep learning-based methods are more general than previously described ones. ManTraNet [221] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localize forgeries in the image. A major limitation of these methods is that they require large training datasets, which are not always available. Furthermore, their performance generally remains dataset-dependent.

In this chapter, we present an improved version of the Noisesniffer method [72], an automatic forgery detector that chases traces of the image noise model disruptions. This method is based on the observation that, even though the camera processing chain modifies the initial raw Poisson-Gaussian noise, the final noise model should be coherent along the image. Local forgeries generally alter this coherence, creating an anomalous noise model in the tampered area [180]. Local anomalies of the noise model, though generally imperceptible to the human eye, can become informative cues for forensic analysis.

4.2 Method

Most non-parametric noise estimation methods share the same principles: they start by selecting the homogeneous regions of the image where noise dominates over the signal, and then they estimate noise in the frequency or the spatial domain using just a small portion of the blocks in the previously selected region [132]. In this work, we follow a similar procedure to identify the blocks within the homogeneous regions that are good candidates for noise estimation. However, instead of using these blocks to estimate the noise level, we are interested in analyzing the spatial distribution of such blocks.

Indeed, if the variance in the homogeneous regions identified in the first step is only explained by noise, the small proportion of blocks that are then used to estimate noise should be a random uniform selection over the homogeneous region. However, if we observe that the blocks used for noise estimation concentrate in a particular part of the homogeneous region, we can deduce that this zone exhibits a suspicious noise deficit.

The proposed method is developed in two steps. Firstly, we compute the set of blocks corresponding to the homogeneous regions and then the subset of those blocks having the lowest standard deviation. Secondly, the spatial distribution of these two sets is compared to detect if there is a statistically significant deviation from one distribution to the other. To do so, we adopt an a contrario approach [56] and assign a number of false alarms (NFA) to each detection.

4.2.1 Distributions computation

The method takes as input an image I of size $X \times Y$, with C color channels. Given w, we first consider all its $w \times w$ overlapping blocks.

When the potential well in the camera sensor is full, the excess of photons does not contribute to the output voltage values and the pixel becomes saturated. Since noise is clipped at saturated pixels, saturated pixels may cause unreliable noise estimations. To avoid this situation, blocks having at least one saturated pixel are discarded. By doing so, we get a list of valid blocks.

When the light interacts with a detector, the photons arrive randomly and independently, resulting in fluctuations in the measured signal. These fluctuations are responsible for the noise at the first step of the image formation process to be signal-dependent. Therefore, to compare the noise levels amongst different blocks, their intensity needs to be taken into account. For each channel, we group the valid blocks in bins with fixed size B according to their mean intensity.

In order to select the suitable regions for noise estimation, for each color channel c and bin b, we first compute the orthogonal Discrete Cosine Transform (DCT II) of each block in the bin, and for each block, we compute its variance in low and medium frequencies. A pair of frequencies (i, j) is said to correspond to a low or medium frequency if $0 < i + j < T$, where T is a fixed threshold that depends on the block size. Note that the pair $(i, j) = (0, 0)$ – the mean of the block term – is not considered [41].

Blocks are then sorted in ascending order according to their variance in low and medium frequencies, and only a small percentile given by the parameter n is kept. These blocks constitute the set L_b^c, where c denotes the corresponding color channel and b the corresponding bin. Since most of the energy corresponding to the image geometry is located at the low and medium frequencies, the selected blocks we keep correspond to the most homogeneous ones.

The intensity variations in these blocks are likely to be explained only by noise. The next step is to compute their variance and order them in ascending order. The subset $V_b^c \subset L_b^c$ corresponds to the $m\%$ of the blocks in L_b^c having the lowest variance, including completely flat blocks (i.e. blocks whose variance is equal to zero). However, if more than the $m\%$ of the blocks in L_b^c are completely flat, the bin is declared invalid. This last precaution aims at preventing false detections due to strong image alterations introduced by the camera image processing pipeline. For instance, when applying strong JPEG compression, artificial flat zones may appear. Indeed, homogeneous regions usually have small DCT coefficients. Therefore, when the quantization factors are big, the JPEG encoder will set them to zero, resulting in flat zones.

Finally, by aggregating the sets obtained for each color channel c and each valid bin b, we define:

$$V = \bigcup_{c} \bigcup_{\text{valid } b} V_b^c \text{ and } L = \bigcup_{c} \bigcup_{\text{valid } b} L_b^c. \tag{4.1}$$

By construction, $V \subset L$.

4.2.2 Statistical validation

Our null hypothesis (H_0) is the absence of any forgery. Under H_0, blocks in V and in L should have the same spatial distribution in the image. Indeed, the blocks in V should correspond to a random and uniform selection over the blocks in L. Nevertheless, when analyzing the distributions of V and L, small deviations from one to the other are likely to happen due to randomness. The fundamental question arises as to whether the observed spatial distribution is likely to happen by chance or not.

A criterion is needed in order to spot deviations that are statistically significant from those that could happen just by chance. Here, we propose to use an a contrario approach [56] to statistically validate these deviations. This theory is based on the Non-Accidentalness Principle, which states that we perceive a structure whenever a large deviation from randomness occurs [8]. However, computing the probability of these events might be hard. This problem is solved by the introduction of the Number of False Alarms (NFA) of an event, which is an upper bound on the expectation of occurrences of such event under the null model [56].

Given a region R to be tested for forgery, our null hypothesis H_0 implies that the blocks[1] in V correspond to a random uniform Poisson point process among the blocks in L. Suppose that for region R there are N blocks $o_1, \dots o_N$ in L, K out of which are also in V. We define the random variables Z_i for $i = 1, \dots, N$ as:

$$Z_i = \begin{cases} 1 & \text{if } o_i \in V, \\ 0 & \text{if } o_i \notin V. \end{cases} \tag{4.2}$$

Since our null model is that the blocks in V correspond to a random and uniform selection over the blocks in L, under this hypothesis the variables Z_i follow a Bernoulli distribution with parameter p, for all $i = 1, \dots N$. Since in each bin and in each channel exactly $m\%$ of the blocks in L_b^c are kept in V_b^c; globally exactly $m\%$ of the blocks of L are in V. Thus, we fix $p = m$.

Following the *a contrario* methodology [56], the number of false alarms (NFA) of the region R is defined as

$$\text{NFA}(R) = N_{\text{tests}} P_{H_0}(Z \geq K), \text{ where } Z = \sum_{i=1}^{N} Z_i. \tag{4.3}$$

The probability $P_{H_0}(Z \geq K)$ is difficult to compute directly because the random variables Z_i with $i = 1, \dots, N$ are not independent. This is because the $w \times w$ blocks used to construct V and L are taken with overlap. We solve this problem by considering that we are making w^2 separate tests: one for each $w \times w$ grid without overlap and assuming that for each of these

[1] In practice, we consider a block o to be in a certain region R if its origin is in R.

tests, we observe N/w^2 blocks in L and K/w^2 blocks in V. Then, the NFA of the region R is defined

$$\text{NFA}(R) = w^2 N_{\text{tests}} \mathcal{B}\left(\frac{K}{w^2}, \frac{N}{w^2}, m\right), \tag{4.4}$$

where N_{tests} is the number of tests to be detailed below and $\mathcal{B}$ denotes the tail of the binomial law:

$$\mathcal{B}(k, n, p) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{4.5}$$

The expression in Equation 4.4 is, in fact, an upper bound of the actual NFA, since at least one of the grids will have more favorable parameters.

A region R is said to be ε-meaningful if $\text{NFA}(R) < \varepsilon$. Once ε is fixed, a region R is detected if it is ε-meaningful. This means that the expected number of regions to be declared ε-meaningful under H_0 is smaller than ε. Therefore, ε gives an *a priori* estimate of the mean number of false detections under H_0. For the rest of the chapter, the threshold ε is set to 1. Although we would normally require a mean number of false detections smaller than 1, due to the discrete nature of the binomial law, the average number of false detections is actually much smaller than the upper bound ε [87].

To complete the formulation we still need to specify the family of regions to be tested. Instead of using rectangular macro-blocks as in the original Noisesniffer formulation [72], we consider more general connected regions, as in [88]. With this aim, we consider a square tessellation of the image, with squares of size $l_\beta \times l_\beta$. Given a block $\beta(i,j)$ of this tilling, where (i,j) denotes the origin of the block, $\beta(i,j)$ is connected with its horizontal and vertical neighbor blocks, namely, to $\beta(i \pm l_\beta, j)$ and $\beta(i, j \pm l_\beta)$. The regions to be tested correspond to those that can be built under this 4-connectivity notion, using the squares of the tessellation as cells. These figures are called polyominoes. The exact number of polyominoes p_n of a given size n is – in general – not known. However, it can be approximated as [103]

$$p_n \approx 0.316915 \times \frac{4.062570^n}{n}. \tag{4.6}$$

Still, we need to consider that each polyomino can be placed at any position in the constructed square tiling. To consider all the possible placements, p_n needs to be multiplied by the number of cell squares, namely $\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$. This is not an exact calculation since it also considers some polyominoes that extend outside the image domain.

Finally, for a given region R of size $|R|$, where the size is the number of cells it contains, the number of tests can be written as:

$$N_{\text{tests}}^{|R|} = \left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right) p_{|R|}. \tag{4.7}$$

Note that we are interested in testing arbitrary-sized regions rather than fixed-sized regions. Hence, we need to distribute the weight of the NFA computed for each region size in such a way that, when computing the final NFA, it truly represents the NFA of the whole image where we have tested several region sizes. To do so, the number of tests is multiplied by the total number of possible region sizes. The region sizes we are interested in are those ranging from one only cell β to half of the image size $\frac{1}{2}\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$, since we only consider forgeries up to this size. Therefore, we have $\frac{1}{2}\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$ possible region sizes.

Then, the number of tests, when considering all the polyominoes sizes we are testing in the whole image is given by

$$N_{\text{tests}} = \frac{1}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right) p_{|R|}. \tag{4.8}$$

With this number of tested regions, the NFA of a candidate region is

$$\text{NFA}(R) = \frac{w^2}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)^2 0.316915 \times \frac{4.062570^{|R|}}{|R|}\mathcal{B}\left(\frac{K}{w^2}, \frac{N}{w^2}, m\right). \tag{4.9}$$

4.2.3 Region growing algorithm

Testing all possible 4-connected regions is computationally intractable. Instead, we propose a heuristic approach to reduce the number of regions that will be evaluated using the *a contrario* approach described in the previous section. The construction of such candidate regions is based on the greedy algorithm proposed by Grompone et al. [88], and the modifications introduced in [202].

The construction can be summarized as follows: a first criterion is used to decide which cells are suspicious of being meaningful. Then, these cells are used as seeds to construct larger regions by iteratively adding connected cells satisfying the region growing criteria. Once the region stops growing, the NFA is computed. If a detection is made, the cells in the region are masked and will not start a new region in further iterations.

For a cell β to be used as a seed, we impose the following criteria. Let N_β be the number of observed blocks in L, of which K_β are also in V. If

$$\frac{K_\beta}{N_\beta} > m, \tag{4.10}$$

then β is a possible seed pixel for a new region. This criterion is based on the fact that the expected proportion of a Binomial law is equal to the probability of success on each Bernoulli experiment, which in our case is equal to m.

Starting from a seed cell, the region-growing algorithm iteratively adds neighbor cells that satisfy the region-growing criterion. To define this criterion, we follow the approach introduced by Tailanian et al. [202]. Namely, in order to decide if a cell is to be added or not to the region, we evaluate the NFA value of the region with and without this cell. If adding the cell makes the region more meaningful (i.e. if it lowers the NFA value), the cell is added.

The region-growing criterion can be stated as follows. Let R be a region and β a neighbor cell. Let N_R and N_β denote the number of observed blocks in L for R and β respectively, of which K_R and K_β are also in V. For the NFA of $R \cup \{\beta\}$ to be smaller than the NFA of R, the following condition must be met:

$$\frac{4.062570}{|R|+1}\mathcal{B}\left(\frac{K_R + K_\beta}{w^2}, \frac{N_R + N_\beta}{w^2}, m\right) < \frac{1}{|R|}\mathcal{B}\left(\frac{K_R}{w^2}, \frac{N_R}{w^2}, m\right). \tag{4.11}$$

4.3 Detailed implementation

As described in Section 4.2.1, the first step of the methods consists in extracting the list of $w \times w$ overlapping valid blocks from image I. Valid blocks are those which do not have any saturated pixels. Here we adopt a relaxed notion of saturation: saturated pixels are those presenting the maximum value or the minimum value in at least one of the image channels.

This notion is more robust to dynamic range changes than just checking pixels with intensities equal to 0 or 255. The implementation of this step is described in Algorithm 2.

Algorithm 2: Computes the indices of the valid blocks (i.e. not containing saturated pixels). (`ComputeValidBlocksIndices`)

```
Input $I$: image of size $X \times Y$ with 3 color-channels
Input $w$: block side
Output validBlocks: list of the indices of the valid blocks

$I_{\text{notsat}} \leftarrow (1)_{X \times Y}$            # initialize all pixels to 1 (not saturated)
for ch $\leftarrow 0$ to $2$ do
    $I_{\text{sat}}^{\text{max}} \leftarrow [I^{\text{ch}} < \max(I^{\text{ch}})]$  # mark as saturated pixels where the max in $ch$ is achieved
    $I_{\text{sat}}^{\text{min}} \leftarrow [I^{\text{ch}} > \min(I^{\text{ch}})]$  # mark as saturated pixels where the min in $ch$ is achieved
    convert $I_{\text{sat}}^{\text{max}}$ and $I_{\text{sat}}^{\text{min}}$ to integers # update saturated pixels
    $I_{\text{notsat}} \leftarrow I_{\text{notsat}} \cdot I_{\text{sat}}^{\text{max}} \cdot I_{\text{sat}}^{\text{min}}$    # where $\cdot$ stands for the Hadamard product
K $\leftarrow (1)_{w \times w}$
$M \leftarrow [I_{\text{notsat}} * \text{K} > w^2 - 0.5]$                # where $*$ stands for convolution
# the result is True only if all the pixels in the $w \times w$ block are
  not saturated
validBlocks $\leftarrow$ indices where M $=$ True
return validBlocks
```

Secondly, the mean intensity of all the $w \times w$ overlapping blocks is computed channel-wise. Though the method only requires to compute the mean intensity of the valid blocks, computing the mean intensities of all the blocks can be efficiently done using a convolution, as described in Algorithm 3.

Algorithm 3: Computes a three dimensional array containing the means of all the $w \times w$ blocks in the image, in each color channel. (`AllImageMeans`)

```
Input $I$: image of size $X \times Y$ with 3 color-channels
Input $w$: block side
Output $I_{\text{means}}$: three dimensional array containing the means of all the $w \times w$
                    blocks in the image, in each color channel

K $\leftarrow \frac{1}{w^2}(1)_{w \times w}$          # define the averaging kernel of size $w \times w$
$I_{\text{means}} \leftarrow I * K$                      # convolution between $I$ and $K$
return $I_{\text{means}}$
```

After these two steps, the channel-wise processing starts. Given a color-channel ch, in order to define the bins, we first sort the valid blocks according to their mean intensity values, as shown in Algorithm 4.

Algorithm 4: Sorts the valid blocks according to their mean intensity in a given color channel ch. (`SortBlocksByMean`)

```
Input ch: color-channel
Input $I_{\text{means}}$: three-dimensional array containing the means of all the w × w blocks
        in the image
Input validBlocks: list of the indices of the valid blocks
Output sortedValidBlocks: list of valid blocks sorted in ascending order according to
        their mean intensity in channel ch.
sortedArgs ← argsort($I_{\text{means}}$[validBlocks, ch])          # sort the blocks
sortedValidBlocks ← validBlocks[sortedArgs]
return sortedValidBlocks
```

Since it is unlikely that the number of valid blocks is a multiple of the number of samples in each bin, given by parameter B, we update this parameter in order to distribute the blocks in bins. This is done according to Algorithm 5.

Algorithm 5: Updates the number of samples per bin. (`UpdateSamplesPerBin`)

```
Input numBlocks: number of valid blocks.
Input $B$: number of samples per bin
Output $\tilde{B}$: updated number of samples per bin
Output numBins: number of bins
numBins ← round(numBlocks/$B$)
if numBins = 0 then
    numBins ← 1                              # force to have at least one bin
$\tilde{B}$ ← $\lfloor$numBlocks/numBins$\rfloor$
return $\tilde{B}$, numBins
```

Then for each bin in channel ch, the first step is to compute the list of blocks that corresponds to the bin b_k, for $k = 0, \ldots, \texttt{numBins} - 1$. All the bins will have $\tilde{B}$ samples except for the last one.

Algorithm 6: Computes the list of blocks corresponding to the k-th bin. (`BinBlockList`)

```
Input $k$: bin.
Input $\tilde{B}$: updated number of samples per bin.
Input numBins: number of bins.
Input numBlocks: number of valid blocks.
Input sortedValidBlocks: list of valid blocks sorted according to their mean.
Output $b_k$: list of the blocks corresponding to the $k$-th bin.
if $k$ = numBins-1 then
    $b_k \leftarrow$ sortedValidBlocks$[($numBins $- 1) \times \tilde{B}, \ldots,$ numBlocks$]$
else
    $b_k \leftarrow$ sortedValidBlocks$[k \times \tilde{B}, \ldots, (k+1) \times \tilde{B}]$
return $b_k$
```

Once the list of blocks corresponding to a certain bin k is computed, we then compute their DCT II. Next, a mask of size $w \times w$ corresponding to the low and medium frequencies is computed, according to Algorithm 7. The thresholds used correspond to those in [41].

Algorithm 7: Computes a mask of size $w \times w$ that corresponds to low-medium frequencies. (`GetTMask`)

```
Input w: block side, $w \in \{3, 5, 8\}$.
Output mask: mask of size $w \times w$ that corresponds to low-medium frequencies.
if $w = 3$ then
    $T \leftarrow 3$
if $w = 5$ then
    $T \leftarrow 5$
if $w = 8$ then
    $T \leftarrow 9$
mask $\leftarrow (0)_{w \times w}$                # define as a zero matrix of sixe $w \times w$
for $i \leftarrow 0$ to $w - 1$ do
    for $j \leftarrow 0$ to $w - 1$ do
        if $i + j \neq 0$ and $i + j < T$ then
            mask$(i, j) \leftarrow 1$
return mask
```

Then, the low-medium frequency energy of each DCT block is computed according to Algorithm 8. For commodity, we use here and afterward the term "variance" to refer to this energy. Indeed, since the expectation is zero, this sum is in fact proportional to the empirical variance.

```
Algorithm 8: Computes the variance in low-medium frequencies of the DCT
II of a $w \times w$ block. (ComputeLowFreqVar)

    Input D: a DCT block of size $w \times w$.
    Input mask: a mask corresponding to low and medium frequencies.      # see
               Algorithm 7
    Output $\sigma^2_{\text{low-med}}$: variance in low-medium frequencies of the DCT II block.

    $\text{D}_{\text{low-med}} \leftarrow D\cdot$ mask   # keep only the low-med frequencies of the block $D$
    $\sigma^2_{\text{low-med}} \leftarrow \sum_{i,j=0}^{w-1} \text{D}_{\text{low-med}}(i,j)^2$
    return $\sigma^2_{\text{low-med}}$
```

Afterwards, a small percentile[2], given by parameter n, of the blocks having the lowest low-medium frequency variance is selected. The selection procedure is summarized in Algorithm 9

```
Algorithm 9: Selects the percentile $n$ of blocks in the bin having the lowest
variances in low-medium frequencies. (SelectBlocksVL)

    Input $\tilde{B}$: updated number of samples per bin.              # see Algorithm 5
    Input $b_k$: list of the blocks in the bin.                        # see Algorithm 6
    Input $\sigma^2_{\text{low-med}}$: list of the variances in low-medium frequencies of the blocks.# see
      Algorithm 8
    Input $n$: a percentile.
    Output $b_k^{\text{low-med}}$: list of the $n \times \tilde{B}$ blocks in the bin having the lowest variances in
                    low-medium frequencies.

    $N \leftarrow n \times \tilde{B}$      # number of blocks that correspond to the $n$ percentile
    sortedArgs $\leftarrow$ argsort$(\sigma^2_{\text{low-med}})$
    $b_k^{\text{sorted}} \leftarrow b_k[\text{sortedArgs}]$
    $b_k^{\text{low-med}} \leftarrow b_k^{\text{sorted}}[0, \ldots, N]$
    return $b_k^{\text{low-med}}$
```

Then, the standard deviation of the selected blocks in the bin is computed. These standard deviations are analyzed as described in Algorithm 10 to check if the bin is valid or not.

```
Algorithm 10: Determines if a bin is valid or not. (BinIsValid)

    Input $\tilde{B}$: updated number of samples per bin.              # see Algorithm 5
    Input $\sigma$: list of the standard deviations of the blocks in the bin.
    Input n: a percentile.
    Input m: a percentile.
    Output binValid: boolean variable, True if bin is valid, False if not.

    $M \leftarrow \tilde{B} \times n \times m$
    binValid $\leftarrow [\text{count}(\sigma = 0) < M]$
    return binValid
```

If the bin is valid, then the m percentile of the blocks having the lowest low-medium frequency variance (selected in Algorithm 9) is kept. The selection procedure is summarized

[2] Throughout this chapter we refer to percentiles and percentages indistinctively.

in Algorithm 11.

Algorithm 11: Selects the percentile m of blocks having the lowest standard deviations amongst those having the lowest low-medium frequency variance. (`SelectBlocksStds`)

```
Input $\tilde{B}$: updated number of samples per bin.            # see Algorithm 5
Input $b^{\text{low-med}}$: list of the blocks with the lowest low-med frequency variance.  # see
              Algorithm 9
Input $\sigma$: list of the standard deviation of the blocks having the lowest low-med
           frequency variance.
Input $n$: percentile of blocks with the lowest energy in low frequencies
Input $m$: percentile of blocks with the lowest standard deviation
Output $b^{\sigma}$: list of the $m \times n \times \tilde{B}$ blocks in the bin having the lowest variances in
            low-medium frequencies.
$M \leftarrow m \times n \times \tilde{B}$ # number of blocks that correspond to the $m$ percentile

sortedArgs $\leftarrow$ argsort($\sigma$)
$b^{\text{low-med}}_{\text{sorted}} \leftarrow b^{\text{low-med}}$[sortedArgs]
$b^{\sigma} \leftarrow b^{\text{low-med}}_{\text{sorted}}[0, \ldots, M]$
return $b^{\sigma}$
```

Finally, if the bin is valid, the n percentile of blocks selected in Algorithm 9 is added to the set L and the m percentile amongst them having the lowest standard deviation selected in Algorithm 11 is added to the set V.

Algorithm 12: Overview of the distributions computation (Section 4.2.1)

```
Input $I$: image of size $X \times Y$ with 3 color-channels
Input $n$: percentile of blocks with the lowest energy in low frequencies
Input $m$: percentile of blocks with the lowest standard deviation
Param $w$: block side
Param $B$: number of samples per bin
Output $V$: set of blocks in the $n$-th percentile having the lowest energy in low
            frequencies
Output $L$: subset of $V$ of blocks in the $m$-th percentile having the lowest standard
            deviation
validBlocks $\leftarrow$ ComputeValidBlocksIndices$(I, w)$
$I_{\text{means}} \leftarrow$ AllImageMeans$(I, w)$
$V \leftarrow \emptyset$
$L \leftarrow \emptyset$
for ch $=$ 0 to 2 do
    $V_{\text{ch}} \leftarrow \emptyset$
    $L_{\text{ch}} \leftarrow \emptyset$
    sortedValidBlocks $\leftarrow$ SortBlocksByMean(ch, $I_{\text{means}}$, validBlocks)
    $\tilde{B} \leftarrow$ UpdateSamplesPerBin(len(validBlocks), $B$)
    numBlocks $\leftarrow$ len(validBlocks)
    numBins $\leftarrow$ round(numBlocks/$\tilde{B}$)
    for $k$ = 0 to numBins- 1 do
        $b_k \leftarrow$ BinBlockList($b, \tilde{B}$, numBins, sortedValidBlocks)
        mask $\leftarrow$ GetTMask$(w)$
        $\sigma^2_{\text{low-med}} \leftarrow$ ComputeLowFreqVar(DCT(block), mask) for all block in $b_k$
        $b_k^{\text{low-med}} \leftarrow$ SelectBlocksVL$(\tilde{B}, b_k, \sigma^2_{\text{low-med}}, n)$
        $\sigma \leftarrow$ [std(block) for all block in $b_k$]
        if BinIsValid$(\tilde{B}, \sigma, n, m)$ then
            $b_k^{\sigma} \leftarrow$ SelectBlocksStds$(\tilde{B}, b_k^{\text{low-med}}, \sigma, n, m)$
            append $b_k^{\text{low-med}}$ to $L_{\text{ch}}$
            append $b_k^{\sigma}$ to $V_{\text{ch}}$
        append $L_{\text{ch}}$ to $L$
        append $V_{\text{ch}}$ to $V$
return $L, V$
```

Once the distributions L and V are computed, they go through the statistical validation process (Section 4.2.2) to spot significant deviations from one another. The NFA computation is described in Algorithm 13.

Algorithm 13: NFA computation

```
Input $I$: image of size $X \times Y$ with 3 color-channels
Input $w$: block side
Input $l_\beta$: block side
Input $m$: percentile of blocks having the lowest standard deviation
Output $V$: set of blocks in the $n$-th percentile having the lowest energy in low
            frequencies
Output $L$: subset of $V$ of blocks in the $m$-th percentile having the lowest standard
            deviation
Output Mask: forgery detection mask

mask $\leftarrow (0)_{X \times Y}$
$\varepsilon \leftarrow 1$
cells $\leftarrow$ list of all the $l_\beta \times l_\beta$ non-overlapping blocks in $I$
for each cell $\in$ cells do
    if cell satisfies seed criteria (Equation 4.10) then
        $R \leftarrow$ [cell]
        $R_{\text{init}} \leftarrow 1$
        $R_{\text{fin}} \leftarrow 0$
        $N_R \leftarrow$ number of $L-$ blocks in $R$
        $K_R \leftarrow$ number of $V-$ blocks in $R$
        while $R_{init} \neq R_{fin}$ do
            $R_{\text{init}} \leftarrow$ len$(R)$
            $R_{\text{fin}} = R_{\text{init}}$
            neighbour cells $\leftarrow$
              list of cells having a common edge to any element in $R$
            for each neighbour $\in$ neighbour cells do
                if neighbour $\notin$
                  $R$ and neighbour satisfies growing criteria (Equation 4.11) then
                    append neighbour to $R$
                    $N_{\text{neighbour}} \leftarrow$ number of $L-$ blocks in neighbour
                    $K_{\text{neighbour}} \leftarrow$ number of $V-$ blocks in neighbour
                    $N_R \leftarrow N_R + N_{\text{neighbour}}$
                    $K_R \leftarrow K_R + K_{\text{neighbour}}$
                    $R_{\text{fin}} \leftarrow R_{\text{fin}} + 1$
        NFA$_R \leftarrow \frac{w^2}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)^2 0.316915 \times \frac{4.062570^{R_{\text{fin}}}}{R_{\text{fin}}} \mathcal{B}\left(\lfloor \frac{K_R}{w^2} \rfloor, \lceil \frac{N_R}{w^2} \rceil, m\right)$
        if NFA$_R < \varepsilon$ then
            mask$[R] \leftarrow 1$
return mask
```

4.4 Experiments

We used the Trace database [13] to evaluate the method. This dataset will be presented in detail in Chapter 6. Still, we summarize here its main characteristics which are relevant to this chapter.

The Trace database is made of 1000 images taken from the Raise dataset [52]. For each image, two forgery masks are made: the *endomask*, obtained by taking a random object from the image's automatic segmentation, and the *exomask*, which is simply the endomask of another image of the set and thus does not correlate to the contents of the image. The concept of the database is to process each raw image with two different pipelines, and splice both processed images according to the forgery masks. Of the six datasets that are proposed, only one is of interest to us: the raw noise level dataset. In this dataset, the two pipelines are the same except for the initial raw noise level model.

In this dataset, noise is added to each raw image before processing it. The noise variance is modeled as $\sigma^2 = A + Bu$, where A and B are constants and u is the noiseless image. Given a base image, the authors sort two different pairs of constants (A_0, B_0) and (A_1, B_1) that determine the two different noise models. Both images - which are in fact the same but having a different noise model - are then processed with the same subsequent pipeline. **Since the proposed method is only able to detect forgeries having lower noise levels, we restrict the analysis to those images**. This sums to 499 images in each, the endomasks and the exomasks datasets. The endomasks dataset is used to find the optimal parameter configuration and to analyze the influence of each parameter on the performance (Section 4.4.1). The comparison with other state-of-the-art methods (Section 4.4.2) was performed on the exomasks dataset, to avoid any kind of overfitting.

To assess the quantitative performance of the different parameters configurations and methods, we used three metrics: the Matthews Correlation Coefficient (MCC), the Intersection over Union score (IoU), and the F1 score, defined by

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}, \tag{4.12}$$

$$IoU = \frac{TP}{TP+FN+FP}, \tag{4.13}$$

$$F1 = \frac{2TP}{2TP+FN+FP}, \tag{4.14}$$

where TP, FP, TN and FN denote the pixel-wise number of true positives, false positives, true negatives, and false negatives, respectively. The IoU and F1 scores vary between 0 and 1. The result is 0 when no true positives are detected and 1 for perfect detection. The MCC score varies from -1 for a detection that is complementary to the ground truth, to 1 for a perfect detection. A score of 0 represents an uninformative result and is the expected performance of any random classifier. The MCC is more representative than the F1 and IoU scores [33, 34], particularly because it is less dependent on the proportion of positives in the ground truth.

The scores were computed for each image and then averaged over each dataset. As most surveyed methods do not provide a binary output but a heat map, to adapt the metrics to the continuous setting, we used their weighted version. In this approach, the value of a heat map H at each pixel x is regarded as the probability of forgery of the pixel. Therefore, given the ground truth mask M, we define the *weighted true positives*, *weighted false positives*, *weighted true negatives* and *weighted false negatives* as

$$TP_w = \sum_x H(x) \cdot M(x), \tag{4.15}$$

$$FP_w = \sum_x (1 - H(x)) \cdot M(x), \tag{4.16}$$

$$TN_w = \sum_x (1 - H(x)) \cdot (1 - M(x)), \tag{4.17}$$

$$FN_w = \sum_x H(x) \cdot (1 - M(x)), \tag{4.18}$$

where · stands for the Hadamard product.

4.4.1 Impact of the parameters in the detection performance

The method has the following parameters:

w: the side length of the blocks used for noise analysis,

B: the number of samples per bin,

n: the percentile of blocks having the lowest variance in low-medium frequencies, stored in L,

m: the percentile of blocks with the lowest standard deviation amongst those having the lowest variance in low-medium frequencies, stored in V,

l_β: the side of the cells used in the NFA computation.

We tested several values for each parameter. For w we tested the values 3, 5 and 8, for B, 20000 and 40000, for n, 0.1, 0.05 and 0.01, for m, 0.5, 0.4, 0.3, 0.2 and 0.1, and for l_β, 60, 80 and 100. The best parameter configurations are given in Table 4.1.

Parameters	MCC	F1	IoU
$w = 3$, $B = 20000$, $n = 0.1$, $m = 0.5$, $l_\beta = 100$	0.3572 (2)	0.3716 (2)	**0.2723** (1)
$w = 3$, $B = 40000$, $n = 0.1$, $m = 0.5$, $l_\beta = 100$	0.3570 (3)	**0.3717** (1)	0.2719 (2)
$w = 3$, $B = 40000$, $n = 0.1$, $m = 0.5$, $l_\beta = 60$	**0.3640** (1)	0.3618 (3)	0.2647 (3)

Table 4.1: Best parameter configurations obtained for the tested values.

We set the optimal parameter configuration to $w = 3$, $b = 20000$, $n = 0.1$, $m = 0.5$ and $l_\beta = 100$, as it delivers the best results when giving the same weight to each of the three metrics. To analyze the individual performance of each parameter, we set the rest of them to the optimal value and only vary the parameter under investigation.

Impact of the block size of the blocks used for noise analysis w

While keeping the rest of the parameters to their optimal value, we tested three possible values for w: 3, 5, and 8. Setting w smaller than 3 would imply too few samples are available to estimate the variance in low and medium frequencies. The results in terms of performance metrics are shown in Table 4.2. We observe that the size of the blocks w highly impacts the performance of the method, being 3 the optimal value among the tested ones.

w	MCC	F1	IoU
3	0.3572	0.3716	0.2723
5	0.2837	0.2879	0.2013
8	0.2039	0.2004	0.1356

Table 4.2: Scores obtained on the endomasks noise level dataset from the Trace database [13] when setting w to 3, 5 and 8, while keeping the rest of the parameters to their optimal value.

Figure 4.1 shows an example of the results obtained when setting w to 3, 5 and 8. Since the forgery is a textured zone, finding homogeneous blocks inside the forgery is more difficult when using bigger blocks, as the distributions show. Therefore, the detection featured when setting w to 3 achieves better results. Indeed, when setting w to 3 the MCC score is 0.9322, the $F1$ score 0.9444, and the IoU score 0.8947 while the corresponding scores when setting w to 5 and 8 are 0.9275, 0.9404, 0.8875 and 0.8730, 0.8965, 0.81239, respectively.

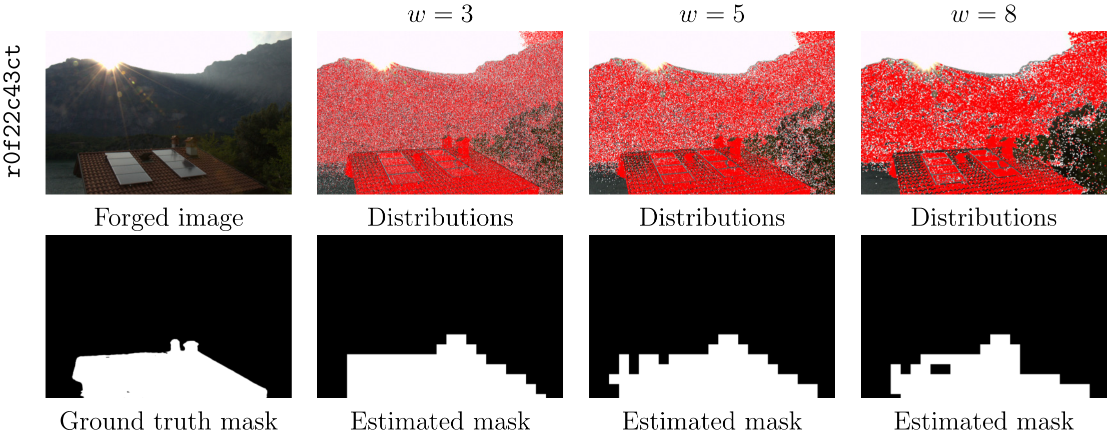


Figure 4.1: Results obtained on image r0f22c43ct from the Trace dataset [13] when setting w to 3, 5, and 8, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red.

Impact of the number of blocks per bin B

While keeping the rest of the parameters to their optimal value, we tested two possible values for B: 20000 and 40000. The results in terms of performance metrics are shown in Table 4.3. We observe that the size of the blocks B has very little influence on the performance.

Indeed, as Figure 4.2 shows, the difference between the results obtained when using bins having 40000 or 20000 samples is very subtle, if any. In this example case, when setting $B = 20000$ the method achieves the scores $MCC = 0.8546$, $F1 = 0.8577$, and $IoU = 0.7509$, while when setting $B = 40000$ the method presents the following scores $MCC = 0.8440$, $F1 = 0.8493$, and $IoU = 0.7381$.

B	MCC	F1	IoU
20000	0.3572	0.3716	0.2723
40000	0.350	0.3717	0.2719

Table 4.3: Scores obtained on endomasks noise level dataset from the Trace database [13] when setting B to 40000 and 20000, while keeping the rest of the parameters to their optimal value.

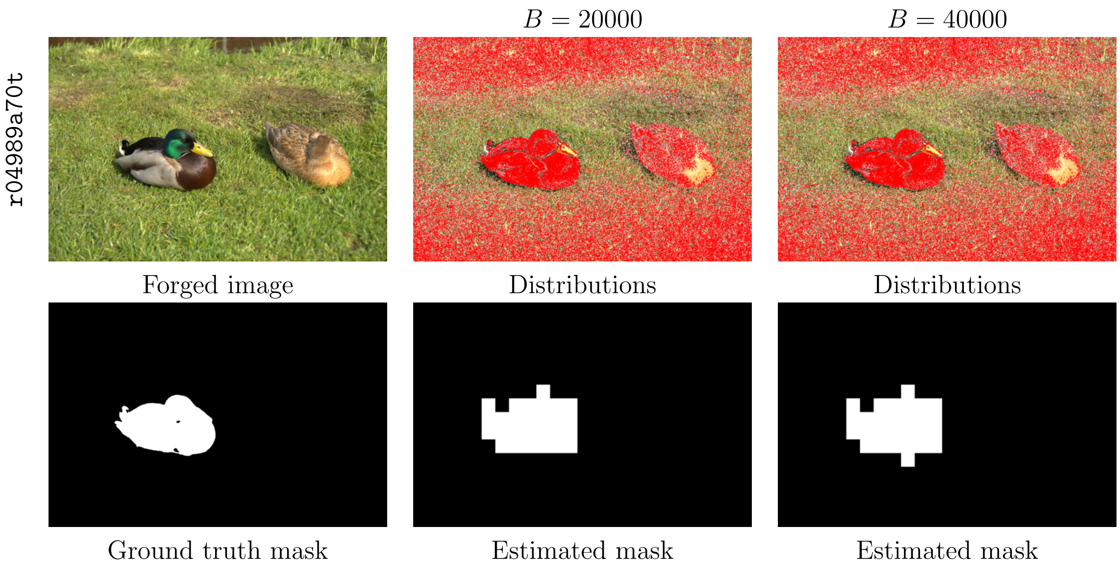


Figure 4.2: Results obtained on image r04989a70t from the Trace dataset [13] when setting B to 40000 and 20000, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. The detection featured when setting $B = 20000$ achieves the scores $MCC = 0.8546$, $F1 = 0.8577$, and $IoU = 0.7509$, while the one featured when setting $B = 40000$ presents the following scores $MCC = 0.8440$, $F1 = 0.8493$, and $IoU = 0.7381$.

Impact of the percentile n

While keeping the rest of the parameters to their optimal value, we tested three possible values for n: 0.1, 0.05, and 0.01. The results in terms of performance metrics are shown in Table 4.4. We observe that the value of the parameter n has a strong influence on the performance of the method. We set 0.1 as an upper bound for n since bigger values could lead to false positives in textured images. Indeed, our method assumes that the n percentile of the total number of $w \times w$ blocks extracted from the image are homogeneous.

Figure 4.3 depicts an example of the different outputs that can be obtained with the method when varying n. When setting $n = 0.1$ the achieved scores are $MCC = 0.7064$, $F1 = 0.7154$ and $IoU = 0.5569$, when setting $n = 0.05$, the scores are $MCC = 0.6564$, $F1 = 0.6675$, $IoU = 0.5009$, finally, when setting $n = 0.01$, the scores are $MCC = 0.5070$, $F1 = 0.5240$, $IoU = 0.3550$.

Regarding the distributions in Figure 4.3, we observe that when setting n to 0.1, the concentration of the blocks in V (painted in red) in the forged zone becomes more evident than in the rest of the cases. As a consequence, the statistical validation step detects more accurately the deviant statistics of the region.

n	MCC	F1	IoU
0.1	0.3572	0.3716	0.2723
0.05	0.3372	0.3382	0.2437
0.01	0.2020	0.1864	0.1295

Table 4.4: Scores obtained on endomasks noise level dataset from the Trace database [13] when setting n to 0.1, 0.05, and 0.01, while keeping the rest of the parameters to their optimal value.

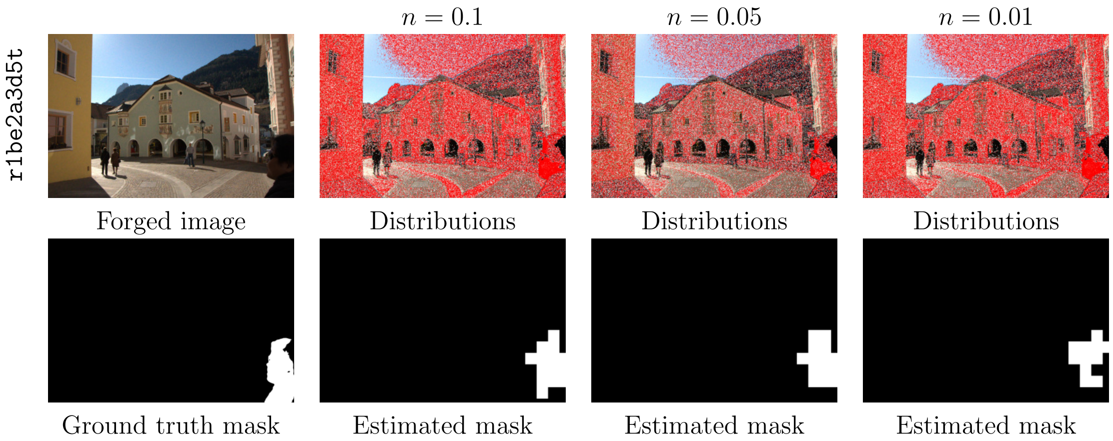


Figure 4.3: Results obtained on image `r1be2a3d5t` from the Trace dataset [13] when setting n to 0.1, 0.05 and 0.01, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. When setting $n = 0.1$ the achieved scores are $MCC = 0.7064$, $F1 = 0.7154$ and $IoU = 0.5569$, when setting $n = 0.05$, the scores are $MCC = 0.6564$, $F1 = 0.6675$, $IoU = 0.5009$, finally, when setting $n = 0.01$, the scores are $MCC = 0.5070$, $F1 = 0.5240$, $IoU = 0.3550$.

Impact of the percentile m

While keeping the rest of the parameters to their optimal value, we tested five possible values for m: 0.1, 0.2, 0.3, 0.4, and 0.5. The results in terms of performance metrics are shown in Table 4.5. The percentile m has a mild but non-negligible influence on the performance of the method.

Figure 4.4 shows an example of the different outputs that can be obtained with the method when varying m. For simplicity, we only display the results for m equal to 0.5, 0.3 and 0.1. When setting $m = 0.5$ the achieved scores are $MCC = 0.7791$, $F1 = 0.8040$ and $IoU = 0.6722$, when setting $m = 0.3$, the scores are $MCC = 0.7344$, $F1 = 0.7643$ and $IoU = 0.6185$, finally, when setting $m = 0.1$, the scores are $MCC = 0.7732$, $F1 = 0.7987$ and $IoU = 0.6648$.

Regarding the distributions depicted in Figure 4.4, we observe that the concentration of blocks in V (painted in red) becomes more evident when setting $m = 0.5$. This makes the detection of the deviant statistics of the region more accurate, as can be seen from the estimated masks. Still, the optimal value of m depends on the size of the forgery: small forgeries are more easily spotted with small m values while large forgeries, such as the one in the example, are better spotted with big values of m.

m	MCC	F1	IoU
0.5	0.3572	0.3716	0.2723
0.4	0.3540	0.3716	0.2723
0.3	0.3498	0.3600	0.2607
0.2	0.3388	0.3437	0.2466
0.1	0.3166	0.3155	0.2234

Table 4.5: Scores obtained on the endomasks noise level dataset from the Trace database [13] when setting m to 0.5, 0.4, 0.3, 0.2 and 0.1, while keeping the rest of the parameters to their optimal value.

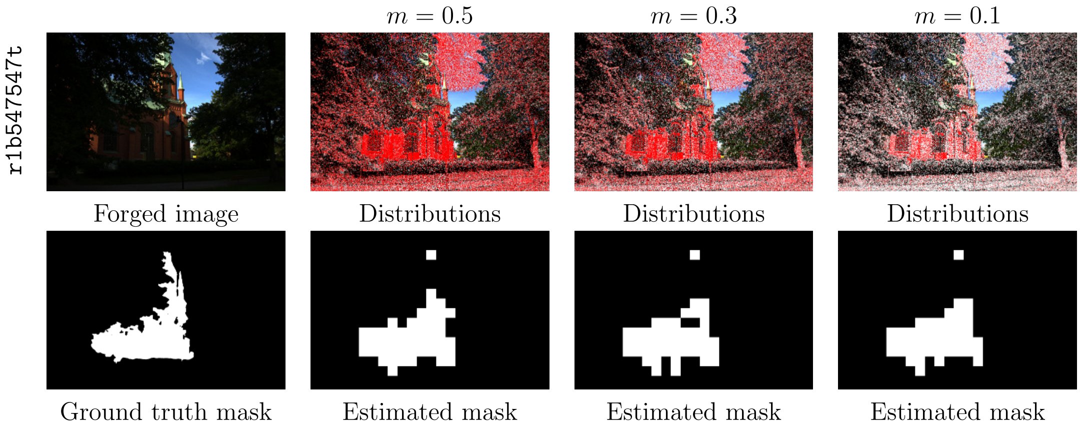


Figure 4.4: Results obtained on image r1b547547t from the Trace dataset [13] when setting m to 0.3, 0.2, and 0.1, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. When setting $m = 0.5$ the achieved scores are $MCC = 0.7791$, $F1 = 0.8040$ and $IoU = 0.6722$, when setting $m = 0.3$, the scores are $MCC = 0.7344$, $F1 = 0.7643$ and $IoU = 0.6185$, finally, when setting $m = 0.1$, the scores are $MCC = 0.7732$, $F1 = 0.7987$ and $IoU = 0.6648$.

Impact of the size of the cells l_β

While keeping the rest of the parameters to their optimal value, we tested three possible values for l_β: 60, 80, 100. The results in terms of performance metrics are shown in Table 4.6.

The parameter l_β has no influence in the distributions computation but it influences the statistical validation step. As can be seen in Table 4.6, the influence of this parameter on the overall performance is not critical. Furthermore, the evidence is not conclusive regarding the optimal value for l_β. Indeed, bigger values deliver higher $F1$ and IoU scores while smaller values deliver better MCC scores.

Figure 4.5 shows an example of the different estimated masks that can be obtained when varying the parameter l_β. Larger zones are detected when setting larger values for l_β. On the other hand, smaller values for l_β seem to capture more accurately the shapes. In terms of scores, the detection featured when setting $l_\beta = 100$ are $MCC = 0.6460$, $F1 = 0.7116$ and $IoU = 0.5523$, when setting $l_\beta = 80$, $MCC = 0.6553$, $F1 = 0.7107$ and $IoU = 0.5512$ and when setting $l_\beta = 60$, $MCC = 0.6473$, $F1 = 0.6945$, $IoU = 0.5320$.

l_β	MCC	F1	IoU
100	0.3572	0.3716	0.2723
80	0.3596	0.3679	0.2705
60	0.3623	0.3604	0.2635

Table 4.6: Scores obtained on the endomasks noise level dataset from the Trace database [13] when setting l_β to 100, 80, and 60, while keeping the rest of the parameters to their optimal value.

4.4.2 Comparison with the state of the art

We used the endomasks dataset to conduct an evaluation comparing the performance of the proposed method (which we shall refer to as Noisesniffer+) to other relevant ones. Among them we compared to classic methods detecting noise traces: the previous formulation of Noisesniffer [72], Lyu [154, 230] and Mahdian [155, 230]; as well as to generic methods

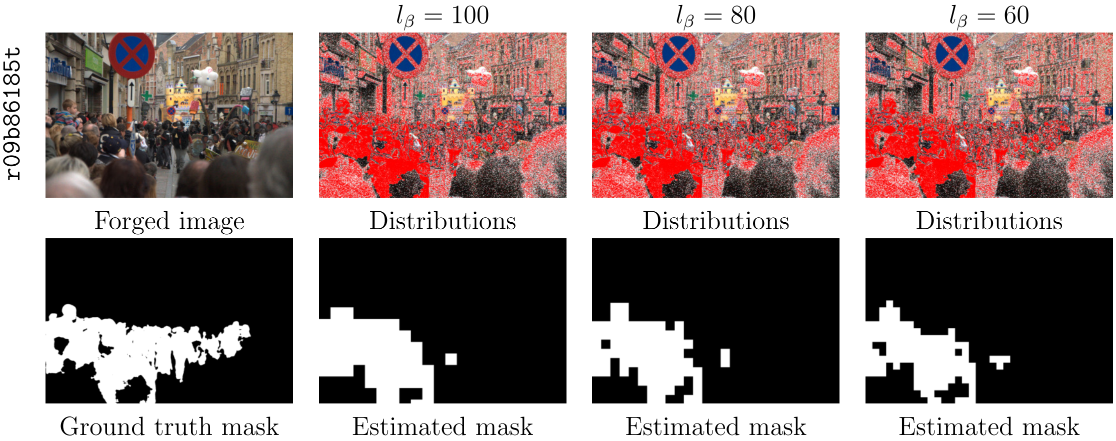


Figure 4.5: Results obtained on image r09b86185t from the Trace dataset [13] when setting l_β to 100, 80, and 60, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. In the three cases, the distributions are the same since the parameter l_β does not affect this step. In terms of scores, the detection featured when setting $l_\beta = 100$ are $MCC = 0.6460$, $F1 = 0.7116$ and $IoU = 0.5523$, when setting $l_\beta = 80$, $MCC = 0.6553$, $F1 = 0.7107$ and $IoU = 0.5512$ and when setting $l_\beta = 60$, $MCC = 0.6473$, $F1 = 0.6945$, $IoU = 0.5320$.

Splicebuster [49], Noiseprint [48], ManTraNet [9, 221]. For each of these methods, we list the source code used for evaluation in Table 4.7.

Method	Ref.	Source Code
Noisesniffer	[72]	https://github.com/marinagardella/Noisesniffer
Mahdian	[155, 230]	https://github.com/MKLab-ITI/image-forensics
Pan	[177, 230]	https://github.com/MKLab-ITI/image-forensics
Splicebuster	[49]	http://www.grip.unina.it/research/83-multimedia_forensics
Noiseprint	[48]	http://www.grip.unina.it/research/83-multimedia_forensics
ManTraNet	[9, 221]	https://www.ipol.im/pub/art/2022/431/

Table 4.7: State-of-the-art methods used for the comparison as well as their reference and link to source code.

In the case of the Lyu and Mahdian algorithms, a specific transformation is needed to score their results. These methods do not act directly as detectors, but rather locally estimate and output the noise level. To turn their outputs into a heat map detection, we computed and normalized the distance of the output to its median. By doing so, we allow these methods to detect regions having noise levels that are far from the median noise level of the image. These regions can have noise deficit or noise excess. The MCC, $F1$ and IoU scores obtained by each of the analyzed methods are presented in Table 4.8.

Firstly, when comparing the performance of Noisesniffer+ in the exomasks dataset (Table 4.8) to the one in the endomasks dataset (Section 4.4.1), we observe that the performance is better when using exomasks. This can be explained by considering that, when using semantically generated masks, the forged zone usually has a homogeneous intensity while when using exogeneous masks, forgeries have more heterogeneous intensities. Since the method performs intra-bin comparisons, the wider range of intensities benefits the detection of the manipulated zone since more bins are involved. This phenomenon has been already reported by Q. Bammey et al. [13] for Noisesniffer.

Method	MCC	F1	IoU
Noisesniffer+	0.502	0.512	0.395
Noisesniffer [72]	0.285	0.274	0.181
Lyu [154, 230]	0.011	0.114	0.066
Mahdian [155, 230]	0.0331	0.114	0.067
Splicebuster [49]	0.178	0.188	0.116
Noiseprint [48]	0.133	0.174	0.106
ManTraNet [9, 221]	0.062	0.110	0.063

Table 4.8: Results of different state-of-the-art forensics tools on the exomasks noise level dataset from the Trace database [13].

Noisesniffer+ outperforms all the evaluated methods. The region growing algorithm presented here, as well as the parameter optimization, doubles the scores obtained by the previous Noisesniffer method [72]. After Noisesniffer, the best-performing methods are Splicebuster and Noiseprint. Splicebuster presents better scores than Noiseprint when using the three proposed metrics. However, the difference is minor for the $F1$ and IoU scores. ManTraNet presents a higher MCC score than the Mahdian and Lyu algorithms. However, their order is inverted for the $F1$ or IoU scores, where the Mahdian and Lyu methods perform equivalently and outperform ManTraNet.

The observed difference in the ranking when considering the MCC score or the $F1$ or IoU scores can be explained by the definition of the scores. The $F1$ and IoU scores neglect the effectiveness of the methods in classifying negative samples. Indeed, their definition excludes the true negatives. On the other hand, the MCC score treats the positive and the negative classes equivalently. Hence, true negatives are as important as true positives. From this analysis, we should expect Splicebuster to have similar detections as Noiseprint but more true negatives. The same applies to ManTraNet, Lyu and Mahdian.

Figure 4.6 shows examples of images where Noisesniffer+ outperforms the other methods. Firstly, we observe that Noisesniffer+ provides more accurate masks than its previous version. For image `r17ad56act` we observe that only a few of the methods timidly highlight the true forgery, except for Noisesniffer+. In most cases, methods highlight the near-saturated regions of the image. On image `r1c5ec853t`, Lyu and Mantranet output very noisy heatmaps, due to the textures of the image. Mahdian, Noiseprint and Splicebuster provide very partial detections of the forgery. As for image `r1bf00696t`, besides Noisesniffer+, only Lyu seems to spot a different behavior on the forged region. Still, their output is tainted with the textures of the image. A similar phenomenon happens for image `r1ea8ccbbt`.

Figure 4.7 shows examples of images where Noisesniffer+ is outperformed by the other methods. For these images, we observe that Noisesniffer+ outputs partial detections (`r10cf67d1t`, `r0fd69c12t` and `r141665bdt`) or less accurate detections (`r059cae86t`). For a detailed analysis of the causes of such limitations, see Section 4.4.3.

For image `r10cf67d1t`, Noisesniffer+ and Noisesniffer are able to detect just a portion of the forgery while Splicebuster, Noiseprint and ManTraNet correctly identify the forged region. Similarly, on image `r0fd69c12t`, our method is outperformed by Mahdian, Splicebuster and Noiseprint. Image `r059cae86t` is correctly detected by Noisesniffer+. However, the provided mask is not as accurate as the ones provided by Noiseprint and Splicebuster. Finally, the partial detection of our method in image `r141665bdt` is outperformed by the detections made by Splicebuster. Other methods such as ManTraNet, Noiseprint and Lyu also spot the forgery. However, they provide noisy heatmaps.

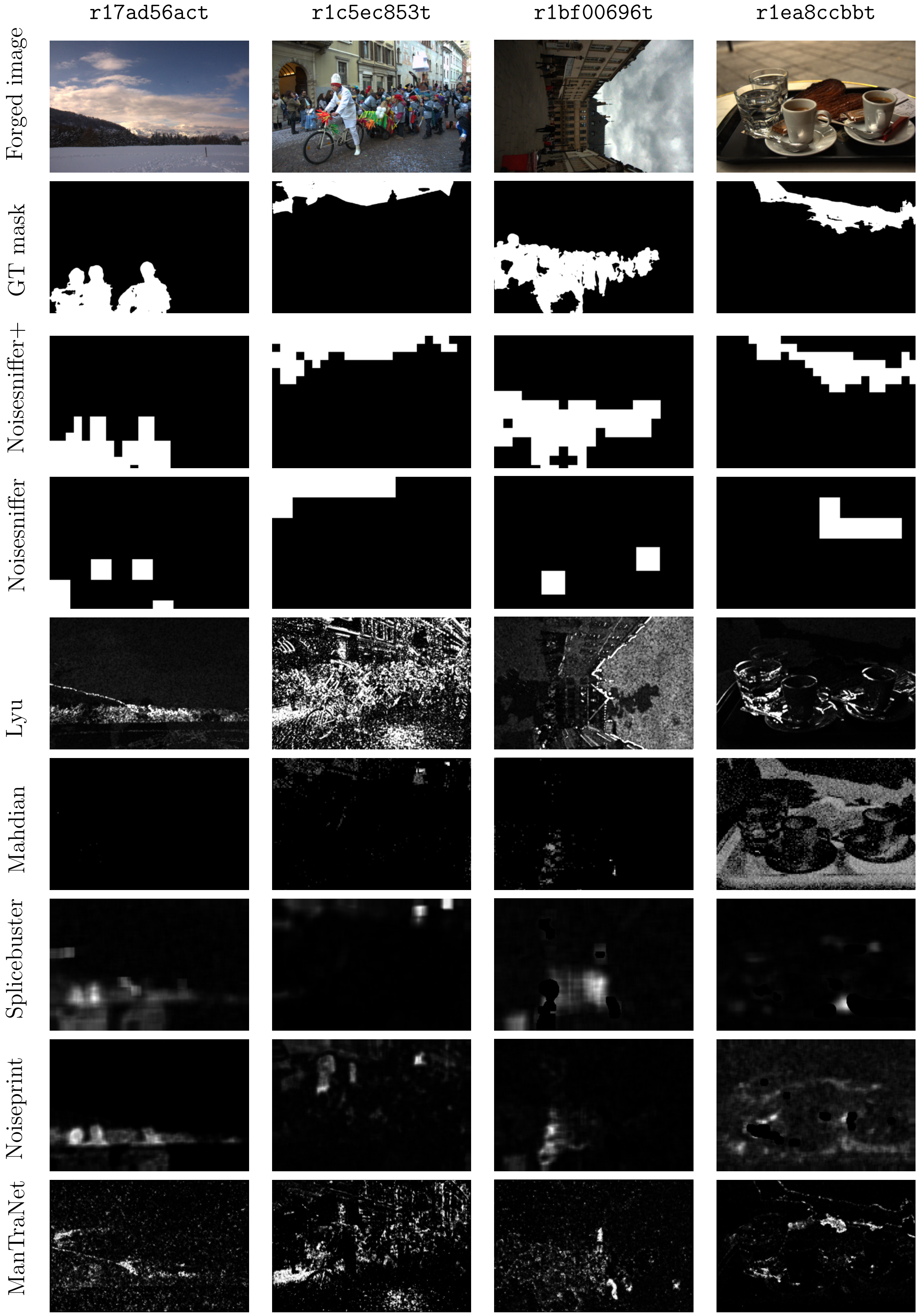


Figure 4.6: Examples from the Trace dataset [13] were the proposed method outperforms the state of the art. For visualization purposes, Splicebuster's and Mahdian's outputs were inverted.

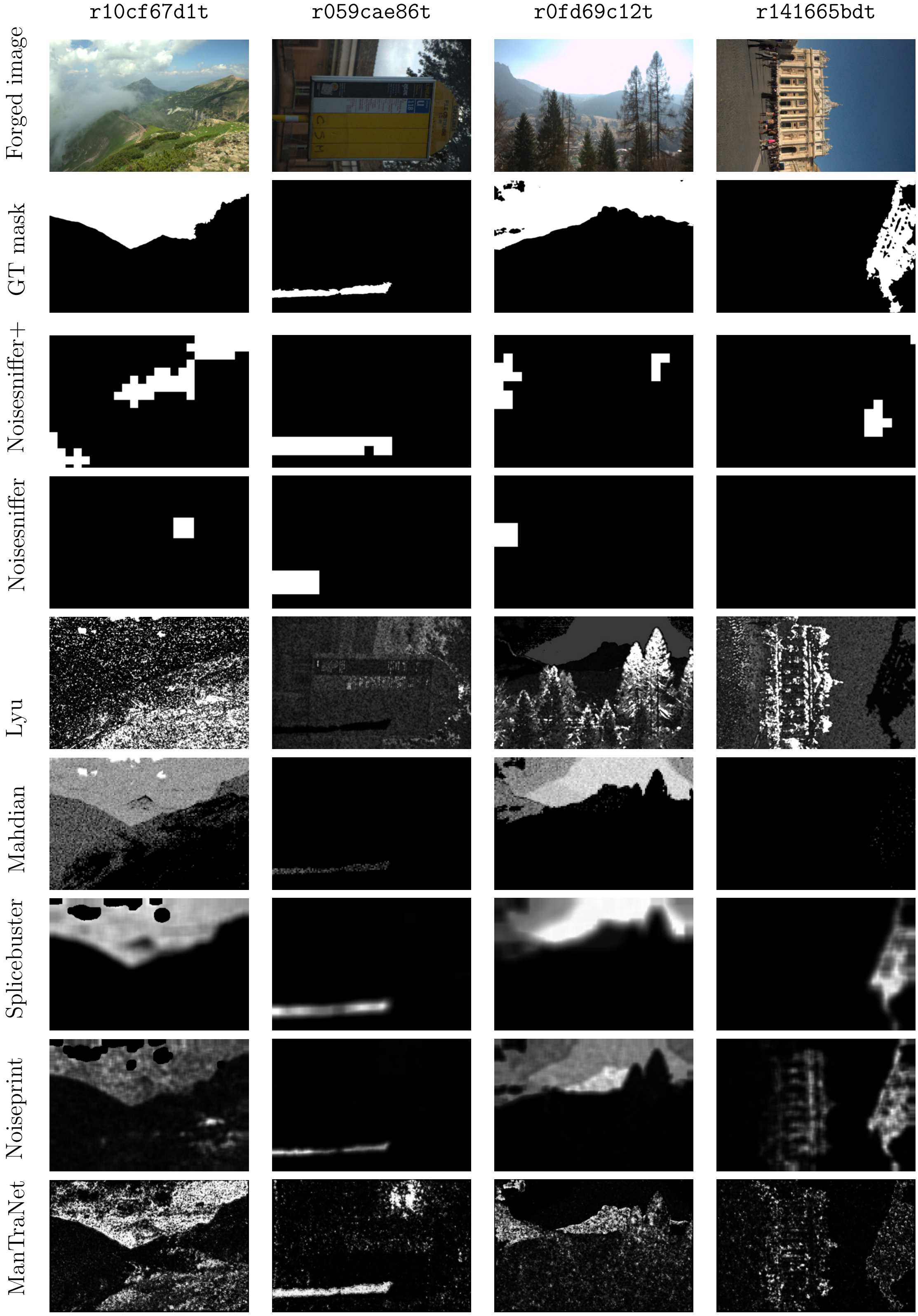


Figure 4.7: Examples from the Trace dataset [13] where the proposed method is outperformed by other the state-of-the-art methods. For visualization purposes, Splicebuster's and Mahdian's outputs were inverted.

4.4.3 Limitations

Besides the cases that are out of the scope of the method, namely when the forgery does not modify the background noise model or has higher noise levels, there are several scenarios under which the method may fail to detect the manipulation. Here, we analyze such cases: in Section 4.4.3 we inspect the missed detections, in Section 4.4.3 we examine the false detections and finally, in Section 4.4.3 we study wrong attributions.

Missed detections

The method may fail to detect forgeries even when the forged area has lower noise levels than the image. The main reasons for this are related to the size of the forgery, the texture of the manipulated area, or even the saturation of the region.

Figure 4.8 shows examples of such cases. Indeed, image `r1b1c1019t` has a very small forgery. Therefore, even if blocks in V concentrate in this region, in the overall spatial distribution this deviation will not be significant. In image `r05db7e7ft`, the forgery is on a textured zone. Since blocks in L -and therefore in V- are chosen from homogeneous areas, only a few blocks are picked in the forged area. Consequently, the method fails to detect it. Finally, in image `r0667a51ft` the forgery is in a saturated zone. Since the method discards blocks having at least one saturated pixel, the method excludes saturated regions from the analysis. Thus, the forgery is undetectable to the method.

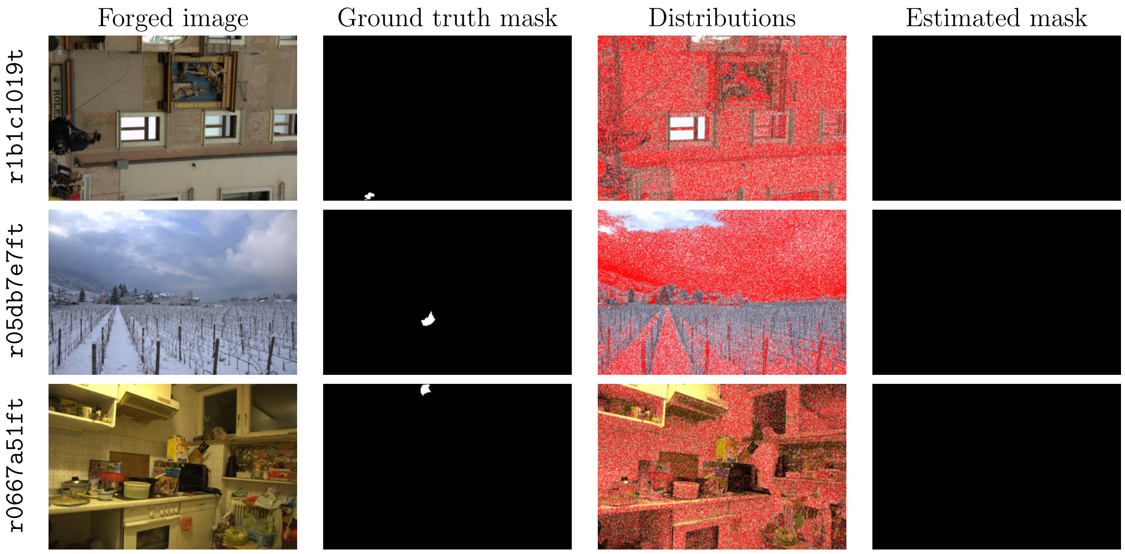


Figure 4.8: Examples of missed detections from the Trace dataset [13]. Even if the forged area has lower noise levels than the background, the method fails to detect them. The main reasons for this to happen are related to the size of the forgery (`r1b1c1019t`), the texture of the manipulated area (`r05db7e7ft`), or even the saturation of the forged region (`r0667a51ft`).

False detections

The main cause of false detections is the presence of flat regions in highly textured images. Since the method assumes that there is a fixed proportion of the image blocks that are homogeneous, if the image is highly textured, some blocks in L will contain texture. Therefore, in the next step, when selecting those blocks having the lowest standard deviation, the method will select the homogeneous ones, since when compared to textured blocks, these blocks present a lower variance.

Figure 4.9 shows an example of false detections in highly textured images. In this case, the method detects the flat zones in the image as forgeries.

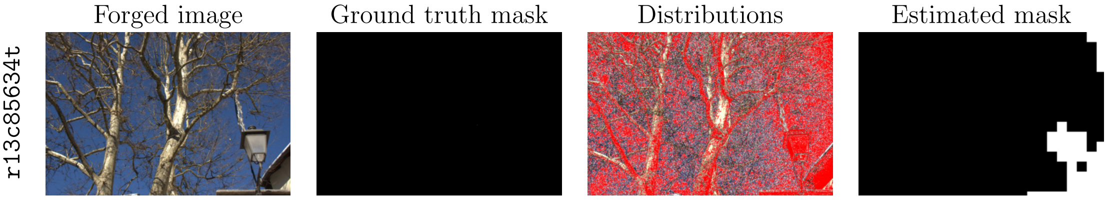


Figure 4.9: Example of false detection from the Trace dataset [13]. The image is highly textured. The method detects the flat zones in the image as forgeries.

Wrong attribution

Although forgeries having higher noise levels than the background are not supposed to be detected by the method nor generate false detections, in some few and very specific cases, this can happen. When the forgery is big enough and increases the noise in most of the blocks having similar intensities, the method identifies as forged the non-forged regions. The wrong attribution phenomenon happens when the method confuses the background model with the one resulting from tampering.

Figure 4.10 shows an example of wrong attribution. In this case, the parameters used are $w = 5$, $B = 40000$, $n = 0.05$, $m = 0.3$ and $l_\beta = 100$. The method detects several forgeries. The one in the rocks is due to the textures in the image, as already explained in Subsection 4.4.3. However, the ones in the sky correspond to the wrong attribution phenomenon described above. Indeed, the forgery covers most of the sky. Therefore, the small pristine zones -that have lower noise levels than the forgery- are regarded as local anomalies and thus detected by the method.

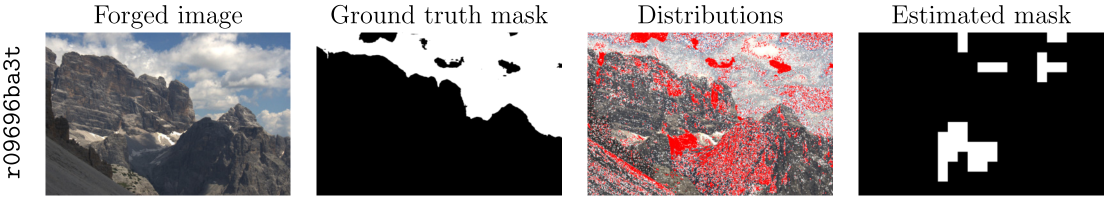


Figure 4.10: Example of wrong attribution from the Trace dataset [13]. The false detection in the rocks is due to the textures in the image, as already explained in Subsection 4.4.3. However, the ones in the sky correspond to the wrong attribution phenomenon. Since the forgery covers most of the sky, the small pristine zones -that have lower noise levels than the forgery- are regarded as local anomalies and thus detected by the method.

4.4.4 Robustness

We focus on two common operations performed on social networks that limit the performance of the method: JPEG compression and downsampling. Since most fake images are shared through these networks, assessing the performance of the method under these manipulations is relevant for real-world usage.

JPEG compression

JPEG compression is the most popular method for lossy compression of digital images [211]. The first step consists in a color space transformation after which the image is tessellated in 8×8 blocks. Each block is then processed by first applying the DCT II and then quantizing the resulting coefficients. The quantization factors depend on the desired final quality, being bigger for lower-quality factors and smaller for higher ones. Finally, the coefficients are lossless compressed.

Since the quantization step mainly acts on the high-frequency coefficients, the distribution computations, in particular the construction of the set L, become strongly affected by this compression scheme. This effect depends, of course, on the quantization factors used. Table 4.9 presents the scores obtained by the method at different compression qualities[3]: $QF = 90$, $QF = 70$ and $QF = 50$. The scores degrade as the compression factor decreases. Even for a quality factor of 90, the method already shows degraded results. Figure 4.11 depicts the effects of JPEG compression at different quality factors (QF) on the performance of the method. The results degrade as the quality factor decreases, and the detected zone is reduced as the compression becomes more aggressive. Besides, strong JPEG compression causes artificial flat zones in the image due to which the method delivers false detections for $QF = 70$ and $QF = 50$.

	MCC	F1	IoU
Uncompressed	0.502	0.512	0.395
$QF = 90$	0.432	0.452	0.337
$QF = 70$	0.318	0.353	0.248
$QF = 50$	0.199	0.249	0.166

Table 4.9: Scores obtained by the proposed method at different JPEG compression levels of the exomasks noise level dataset from the Trace database [13]. The scores degrade as the quality factor decreases.

Image downsampling

According to the Nyquist sampling theorem, resampling an image with a lower than twice the maximum frequency produces aliasing artifacts. These artifacts can be avoided by applying a low-pass filter to the image before resampling. Although there is no standard way of downscaling an image, here we cover two possible approaches: first, a naive resampling where no pre-filtering is applied, and a more sophisticated case where a Gaussian blur is applied before resampling.

Table 4.10 presents the scores obtained by the method when scaling the image to half its original resolution with and without pre-filtering the image before downsampling. The downsampling applied consists simply at keeping one pixel out of two in each direction. The Gaussian pre-filtering consists in convolving the image with a Gaussian kernel of size 5×5 and standard deviation $(1.2, 1.2)$.

Firstly, we observe that the scores degrade when the image is downsampled. Downsampling an image with or without pre-filtering reduces the power of the method since fewer samples are available to estimate the distributions. Therefore, detecting statistically significant deviations becomes more difficult. Furthermore, since the size of the cells used for the region-growing

[3] Images where compressed using the 'ImageMagick' (`https://imagemagick.org`) `convert` command line and specifying the corresponding quality factor with the flag `- quality`.

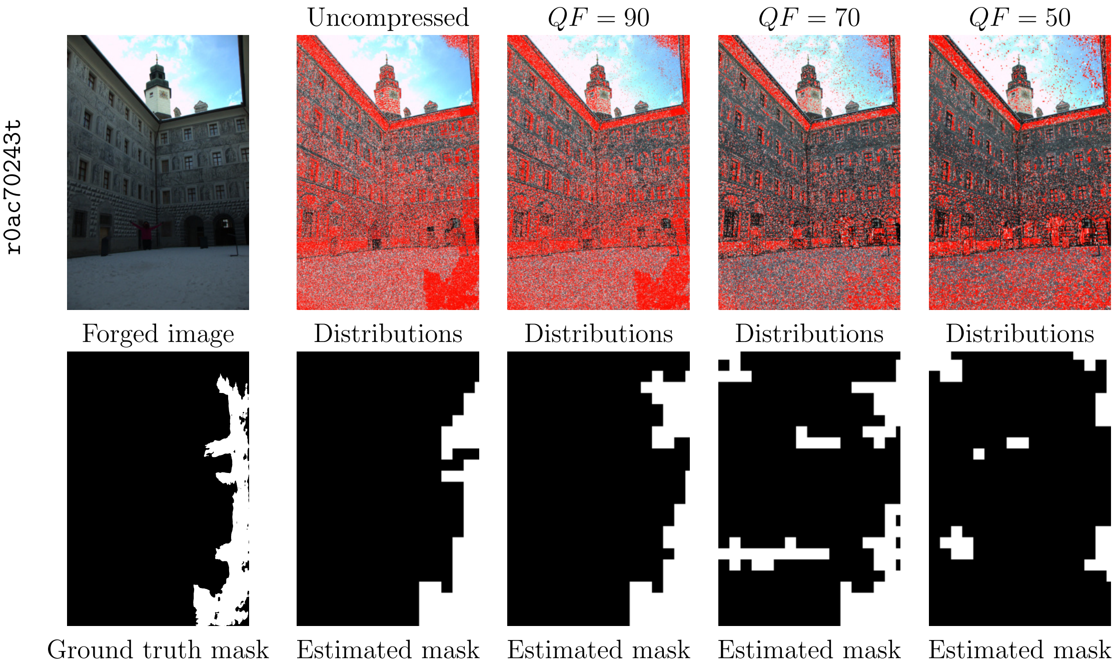


Figure 4.11: Results obtained on image r0ac70243t from the Trace dataset [13] when compressed at different quality factors. The results degrade as the quality factor decreases.

algorithm is fixed, this entails a mask accuracy loss; the masks do not adapt correctly to irregular borders.

Gaussian pre-filtering is equivalent to applying a low-pass filter to the image in the frequency domain. Similarly to JPEG compression, the cancellation of high-frequency coefficients strongly affects the construction of the sets L and V. Therefore, when Gaussian pre-filtering is applied before downscaling, the results are worse than when downscaling the image itself.

Figure 4.12 presents an example of the effects of downscaling with and without pre-filtering for a scaling factor $s = 0.5$. We can observe the reduction in the number of samples in the sets L and V for both downsampling scenarios. When downscaling is applied without any prefiltering, the method is able to detect most of the forgery. However, the mask accuracy is affected by the smaller image resolution. When pre-filtering the image before downscaling, the method is also able to detect most of the forgery but looses some small regions due to the influence of the Gaussian blur in the high frequencies.

	MCC	F1	IoU
Original resolution	0.502	0.512	0.395
Downsampling ($s = 0.5$) without prefiltering	0.423	0.450	0.340
Downsampling ($s = 0.5$) with Gaussian blur prefiltering	0.399	0.430	0.3238

Table 4.10: Scores obtained by the proposed method at a scaling factor $s = 0.5$ with and without pre-filtering on the exomasks noise level dataset from the Trace database [13]. The performance of the method degrades when the image is downscaled. This degradation is worsen when Gaussian pre-filtering is applied.

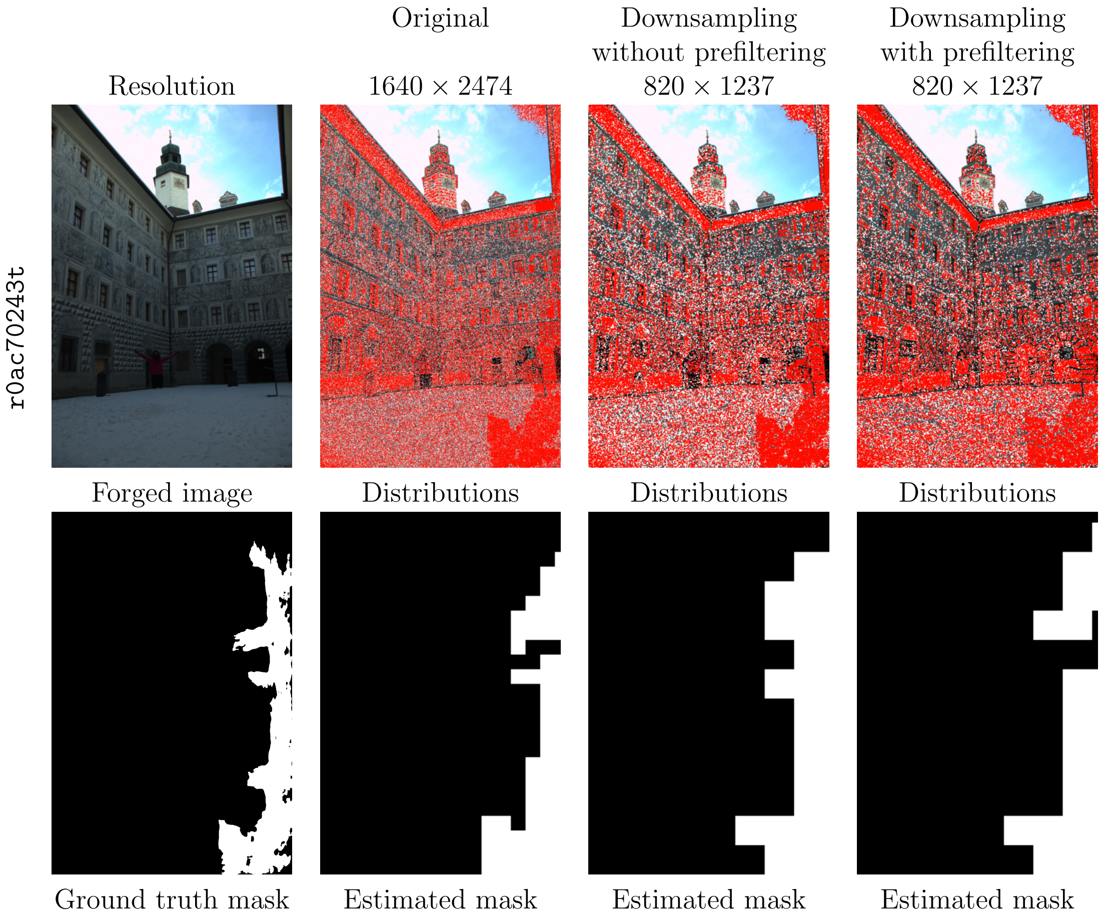


Figure 4.12: Results obtained on image r0ac70243t from the Trace dataset [13] when downscaled at a scaling factor $s = 0.5$ with and without pre-filtering. The forgery is detected when downscaling is applied without any pre-filtering. However, when Gaussian blur is applied prior to downscaling, the method is unable to perform any detection.

4.5 Conclusion

This chapter describes and improves Noisesniffer, a forgery detection method based on noise analysis. As a relevant addition to other existing noise-based methods, Noisesniffer incorporates a statistical validation step detecting only the inconsistencies that could not happen by chance. Each detection is associated with a number of false alarms (NFA). In this chapter, we build on the previous statistical detection step to provide refined detections. The method also provides, together with the statistically validated detection mask, a visual exploration of the relative flat patch distributions which can also aid the interpretation of the results. Results show that the proposed modifications to the original Noisesniffer method improve its performance. Additionally, the method outperforms the state of the art on forgeries having noise deficit.

Chapter 5

Exploring Image Forgery Detection via Forensic Similarity Graphs

As mentioned in Chapter 2, sniffing the traces left by the camera processing chain can be exploited for forgery detection. Most classic methods are constructed over hand-crafted features. However, learning camera related features has also been attempted in literature. In the article "Exposing Fake Images with Forensic Similarity Graphs", O. Mayer and M. C. Stamm introduce a novel image forgery detection method. The proposed method is built on a graph-based representation of images, where image patches are represented as the vertices of the graph, and the edge weights are assigned in order to reflect the forensic similarity between the connected patches. In this representation, forged regions form highly connected subgraphs. Therefore, forgery detection and localization can be cast as a cluster analysis problem on the similarity graph. In this chapter, we present briefly the method and offer an online executable version allowing everyone to test it on their own suspicious images. The work presented here is published as *Image Forgery Detection via Forensic Similarity Graphs* in IPOL [70] and the online demo is available at: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=432.

5.1 Introduction

The forgery detection methods introduced in the two previous chapters (Chapter 3 and Chapter 4) tackle imperceptible specific traces that are left in the image at each step of the camera processing pipeline [180]. Namely, they search for inconsistencies in the noise model. Indeed, several methods share this approach and are designed to analyse specific cues such as demosaicing inconsistencies [10, 35], JPEG artifacts [176, 225] and more [97].

However, this is not the only approach that has been introduced in literature. As mentioned in the previous chapters, deep learning methods for forgery detection have also been developed. Some of these methods are directly trained on forged images and aim at identifying specific manipulations [160, 190, 221]. However, training a network to learn all possible manipulations is not feasible.

Following this observation, other deep learning approaches have been proposed [23, 48, 90, 99, 158, 163]. These methods mimic the hand crafted methodology adopted by classical methods. Namely, they aim at extracting features related to the camera processing chain, and to detect local inconsistencies in these features. The main difference is that, in this case, these features are learnt.

In particular, Mayer and Stamm [162] show the forensic potential of features learnt for source camera classification. Indeed, the authors develop the “forensic similarity score" that aims at distinguishing if two image patches share the same forensic traces or not. In this chapter, we explore the approach in [163] which exploits this forensic similarity measure for forgery detection.

5.2 Forensic similarity score

5.2.1 Problem formulation

The problem can be stated as follows: Given two image patches, we want to assign a score of 0 to the pair of patches if they have different forensic traces, and a score of 1 if they share the same forensic traces. That is, we search for a map $C : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ is the space of all image patches, such that

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } X_1, X_2 \text{ have different forensic traces,} \\ 1 & \text{if } X_1, X_2 \text{ have the same forensic traces.} \end{cases}$$

This problem can be tackled in three steps. First, a suitable set of N features capturing the forensic information is extracted from each patch, by means of a feature extractor $f : \mathcal{X} \to \mathbb{R}^N$. The resulting feature vectors are then compared based on a similarity function $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$. Finally, the similarity measure is compared to a threshold τ so as to obtain a binary output. The map C can be then written as

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } S(f(X_1), f(X_2)) \leq \tau, \\ 1 & \text{if } S(f(X_1), f(X_2)) \geq \tau. \end{cases}$$

This way, the problem of finding C amounts to find two functions $f : \mathcal{X} \to \mathbb{R}^N$ and $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$ such that $S(f(X_1), f(X_2))$ is as close to 0 as possible whenever X_1 and X_2 have different forensic traces, and as close as possible to 1 whenever the two image patches share the same traces. Figure 5.2 shows the system overview.

5.2.2 Method

In [162], Mayer and Stamm propose to design both the feature extractor function f and the similarity function S based on a learning strategy. In this section we specify the architecture as well as the training strategies developed in their work.

Architecture

The feature extractor is based on the MISLnet architecture [17] and is depicted in Figure 5.1. Namely, it consists of 5 convolutional blocks and 2 fully connected layers. Each of the convolutional layers, except for the first one, is followed by batch normalization, hyperbolic tangent activation and max-pooling. The size of the convolutional filters used at each layer are $5 \times 5 \times 3 \times 6$, $7 \times 7 \times 6 \times 96$, $5 \times 5 \times 96 \times 64$, $5 \times 5 \times 64 \times 64$ and $1 \times 1 \times 64 \times 128$ respectively. A stride equal to 1 is used in all the layers except for the second one, where the stride is set to 2. The last convolutional layer, which uses 1×1 kernels, can be regarded as a learned cross-feature maps associations. The max-pooling operation is performed using 3×3 kernels. The two fully connected layers that follow the convolutional blocks consist both of 200 neurons with hyperbolic tangent activation.

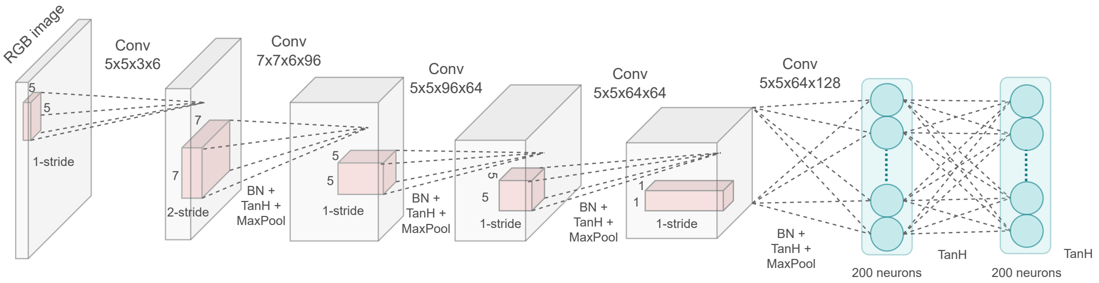


Figure 5.1: Feature extractor architecture.

Two such feature extractor networks, in siamese configuration with weight-sharing, are used to process in parallel the two patches, producing a feature vector for each patch. Then, a similarity network takes both feature vectors as input and computes their similarity score. Figure 5.2 shows the complete system overview. The first layer of the similarity network consists of 2048 neurons with ReLu activation, that maps each feature vector into a new feature space. The authors use a hard-sharing siamese configuration for this first layer. Then, a new feature vector is constructed by concatenating both feature vectors and their element-wise multiplication. This vector feeds another fully-connected layer with 64 neurons. Finally, a single-neuron layer with sigmoid activation takes the resulting 64-dimensional vector and produces the similarity score associated to the pair of input patches.[1]

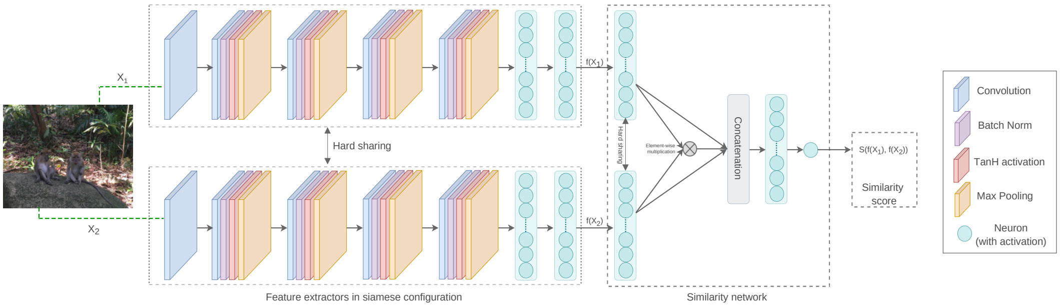


Figure 5.2: System overview. The first module consists of a pair of feature extractor networks in siamese configuration with weight-sharing. It takes two image patches and computes its corresponding feature vectors. These vectors are then compared by means of the second module (the similarity network), which computes a similarity score associated to the pair of image patches.

Dataset

Mayer and Stamm collected a dataset of 47,785 images from 95 different camera models. Among them, 26 camera models come from the Dresden dataset [81] while the rest are from the authors' database. This dataset is divided into three disjoint subsets. Subset 1 consists of 50 camera models selected randomly from those having at least 40,000 non overlapping 256×256 patches. Subset 2 comprises 30 camera models from the remaining ones having at least 25,000 non-overlapping patches. Subset 3 comprises the remaining 15 camera models.

[1] In practice, the authors use two output units with softmax activation, one indicating "similar" traces and the other one indicating "different" traces. The observed output for evaluation is the one corresponding to "similar".

The complete list of camera models is given in Table 5.1. The camera models in blue were collected from the Dresden dataset [81].

Subsets 1 and 2 are used for training (see Section 5.2.2) while Subset 3 is used for evaluation, which we will not cover here. The interested reader is referred to the original paper [162].

Subset 1			
Apple iPhone 4	Agfa Sensor530s	Apple iPhone 4s	Canon EOS SL1
Apple iPhone 5	Canon PC1730	Apple iPhone 5s	Canon A580
Apple iPhone 6	Canon ELPH 160	Apple iPhone 6+	Canon S100
Apple iPhone 6s	Canon SX530 HS	Canon SX420 IS	Canon SX610 HS
Casio EX-Z150	Fujifilm S8600	Huawei Honor 5x	LG G2
LG G3	LG Nexus 5x	Motorola Maxx	Motorola Turbo
Motorola X	Motorola XT1060	Nikon S33	Nikon S7000
Nikon S710	Nikon D200	Nikon D3200	Nikon D7100
Panasonic DMC-FZ50	Panasonic FZ200	Pentax K-7	Pentax OptioA40
Praktica DCZ5.9	Ricoh GX100	Rollei RCP-7325XS	Samsung Note4
Samsung S2	Samsung S4	Samsung L74wide	Samsung NV15
Sony DSC-H300	Sony DSC-W800	Sony DSC-WX350	Sony DSC-H50
Sony DSC-T77	Sony NEX-5TL		
Subset 2			
Apple iPad Air 2	Blackberry Leap	Apple iPhone 5c	Canon Ixus70
Agfa DC-733s	Canon PC1234	Agfa DC-830i	Canon G10
Canon SX400 IS	Canon T4i	Fujifilm XP80	Fujifilm J50
HTC One M7	Kodak C813	Kodak M1063	LG Nexus 5
Motorola Nexus 6	Nikon D70	Nikon D7000	Nokia Lumia 920
Olympus TG-860	Panasonic TS30	Pentax OptioW60	Samsung Note3
Samsung Note5	Samsung S3	Samsung S5	Samsung S7
Sony A6000	Sony DSC-W170		
Subset 3			
Agfa DC-504	Canon Ixus55	Agfa Sensor505x	Canon A640
Canon Rebel T3i	LG Optimus L90	LG Realm	Nikon S3700
Nikon D3000	Olympus 1050SW	Samsung Lite	Samsung Nexus
Samsung Note2	Samsung S6	EdgeSony DSC-T70	

Table 5.1: Camera models used for training and evaluation. Camera models in blue come from the Dresden dataset [81]. Subset 1 is used during the first training phase. Subset 1 and 2 are used during the second training phase. Subset 3 is used for evaluation.

Training procedure

The system is trained in two phases. In the first phase, the feature extractor is trained by adding a fully connected layer with softmax activation. The feature extractor is trained as a source camera classifier, using image patches with associated labels corresponding to their source camera model. Research indicates that the deep features associated to camera model classification provide a good starting point for several forensics tasks [164]. The authors use a cross-entropy loss, optimized using stochastic gradient descent for 30 epochs, with batches of 50 images. Initially the learning rate is set to 0.001, and is halved every three epochs. The authors train two versions of the feature extractor: one using 128×128 patches and another one using 256×256 patches.

During the second training phase, the similarity network is trained to target a specific task. Here, the task is to determine if two image patches come from the same camera model, but it could be to determine a specific editing operation or a specific parameter given an editing operation. The labels in the training dataset are assigned accordingly: 0 indicates that the patches come from the same camera model, and a 1 indicates they come from different ones. During this phase, the weights of the feature extractor are fine-tuned, i.e. they are also updated to better fit the particular task. The similarity network is trained using stochastic gradient descent with cross-entropy loss for 30 epochs. The learning rate is initialized to 0.005, and then is halved every three epochs.

During the first training phase, the feature extractor is trained using 40,000 randomly sampled image patches from each of the camera models in Subset 1, giving a total of 2,000,000 image patches. During the second training phase, the similarity network is trained and the feature extractor weights are updated using a training dataset of pairs of image patches. This dataset is constructed using camera models in Subset 1 and Subset 2.

5.3 Forensic similarity graph for forgery detection

A similarity graph is a graph where the edge weights represent the similarity between the connected nodes. In [163], Mayer and Stamm propose to construct a similarity graph from the image using a similarity measure that reflects the similarity of the processing pipeline. To do so, they represent the image patches as nodes and assign weights to the edges according to the forensic similarity score between them, described in Section 5.2. The resulting graph is a fully connected graph, where edges connecting patches having similar forensic traces will have weights near 1, and edges connecting patches with low forensic similarity will have weights near 0.

In this representation, patches that have undergone the same processing operations are expected to form communities. Communities are characterized by strong connections within the members and weak connections to non-members. This way, forgery detection can be formulated as a community detection problem, and forgery localization as a community partition problem. Several community detections techniques have been reported in literature [69]. In their work, Mayer and Stamm focus on two particular techniques: spectral clustering and modularity optimization. However, it is important to notice that it is straightforward to extend the proposed approach to other clustering methods.

Spectral Clustering

A weighted graph $G = (V, W)$ is a pair where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and W is a symmetric matrix satisfying $W_{ij} \geq 0$ for all $i, j = 1, \ldots, n$ and $W_{ii} = 0$ for all $i = 1, \ldots, n$. The weight matrix W can be considered as an extension of the adjacency matrix for non-weighted graphs. Indeed, if $W_{ij} \in \{0, 1\}$, both definitions become equivalent.

Spectral clustering methods aim at partitioning the graph G based on the spectrum of W itself or any other matrices built on it. In this work, the authors focus on the study of the spectrum of the graph Laplacian matrix, defined as

$$L = D - W, \tag{5.1}$$

where D is the degree matrix defined as $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ if $i \neq j$. The matrix D generalizes the degree matrix for non-weighted graphs. Indeed, if $W_{ij} \in \{0, 1\}$, for all i, j, both definitions are equivalent.

Since, by construction, all the rows of L sum up to 0, $\lambda_1 = 0$ is always an eigenvalue for L, corresponding to the eigenvector $(1, 1, \ldots, 1)$. Furthermore, its multiplicity corresponds to the number of connected components in the graph. Indeed, the membership vectors (i.e. vectors having ones for the connected nodes and zeros for the other nodes) will be eigenvectors associated to the 0 eigenvalue. This property of the graph Laplacian can be applied to community structure detection, by observing that if there are weakly linked sub-graphs, the smallest non-zero eigenvalue will still be close to zero [66].

The authors use the *eigengap heuristic* [153] to detect if more than one community exists by computing the second smallest eigenvalue λ_2, and comparing it to a pre-defined threshold $\tau = 100$, derived empirically. If λ_2 is smaller than τ, the image is classified as forged and if it is bigger, as non-forged.

In case the image is considered as tampered, forgery localization can be performed by finding the community partition. The authors only consider the case when two communities exist. In this case, graph bipartition can be achieved from the eigenvector of the second smallest eigenvalue λ_2 of the graph Laplacian matrix [66]. To do so, it is enough to compute an eigenvector u_2 associated to λ_2 and partition the graph according to the sign of each component of u_2. Indeed, for each vertex v_i, labels are assigned according to:

$$c_i = \begin{cases} 1 & \text{if } u_2^i \geq 0 \\ 0 & \text{if } u_2^i < 0, \end{cases} \tag{5.2}$$

where c_i denotes the predicted community for the node v_i.

Modularity optimization

Modularity was introduced as a measure of the quality of a particular graph partition [171]. It is built on the observation that random graphs are not expected to have communities structures. Therefore, by comparing the observed edge density within a community to the expected edge density given by the background model, one can assess the meaningfulness of the given community structure. Modularity can be expressed as

$$Q = \frac{1}{2m} \sum_{i,j} (W_{ij} - E(W_{ij}))\, \delta(c_i, c_j), \tag{5.3}$$

where the sum is taken over all the pair of vertices, m is the weighted total number of edges $m = \frac{\sum_{i,j} W_{ij}}{2}$, W is the weights matrix, $E(W_{ij})$ is the expected weight for an edge connecting vertices i and j given by the background model, δ is the Kronecker δ-function and c_i is the community to which vertex i belongs.

The choice of the null model is not entirely unconstrained according to Newman [172]. Firstly, $E(W_{ij})$ should be equal to $E(W_{ji})$ since we are considering undirected graphs. Secondly, Q should be equal to 0 when all the vertices belong to a single community. Setting $c_i = c_j$ for all $i, j = 1, \ldots, n$ it follows that

$$\sum_{i,j} W_{ij} = \sum_{i,j} E(W_{ij}) = 2m. \tag{5.4}$$

Despite these constraints, there are several possible background models satisfying these conditions. Mayer and Stamm adopt the same null model as Newman [172]. Firstly, this model imposes that the expected degree of each vertex is equal to the actual degree in the graph,

$$\sum_i E(W_{ij}) = D_{jj}. \tag{5.5}$$

This condition implies automatically the constraint $\sum_{i,j} E(W_{ij}) = 2m$. Secondly, the null model suggested by Newman imposes that the expected weight of an edge connecting vertices i and j is the product of a function f of their degrees

$$E(W_{ij}) = f(D_{ii})f(D_{jj}). \tag{5.6}$$

Equation 5.5 and Equation 5.6 imply

$$E(W_{ij}) = \frac{D_{ii}D_{jj}}{2m}. \tag{5.7}$$

Therefore, under this background model, modularity can be written as

$$Q = \frac{1}{2m}\sum_{i,j}\left(W_{ij} - \frac{D_{ii}D_{jj}}{2m}\right)\delta(c_i, c_j). \tag{5.8}$$

Modularity optimization aims at finding the community partition for which the modularity is maximized, i.e.

$$Q^{\mathsf{opt}} = \max_{c_1,\ldots,c_n} \frac{1}{2m}\sum_{i,j}\left(W_{ij} - \frac{D_{ii}D_{jj}}{2m}\right)\delta(c_i, c_j). \tag{5.9}$$

Different strategies to optimize the modularity have been addressed in the literature [69]. Mayer and Stamm use the fast greedy technique introduced in [38], where vertices are successively joined to form larger communities in a way such that modularity increases.

If the optimal value for the modularity is close to 0, there is no evidence of community structure [171]. Therefore, to detect if an image is tampered the authors compare Q^{opt} to a pre-defined threshold $\tau = 0.025$. Then, if the observed Q^{opt} is larger than τ, the image is classified as forged; otherwise, the image is classified as pristine.

In case the image is classified as forged, the same optimization problem can be used to partition the image into clusters. Indeed, in that case the optimization problem can be expressed as

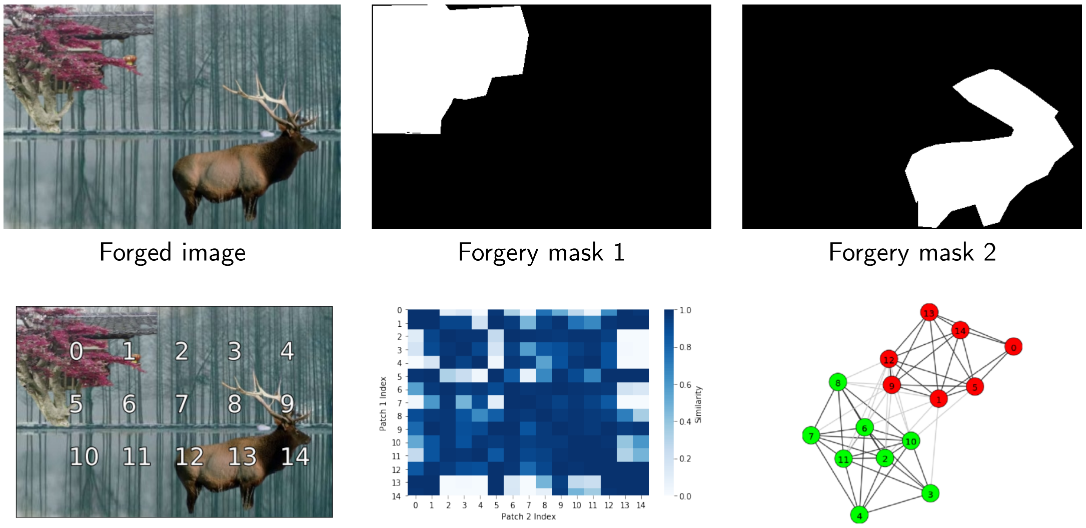
$$c_1^{\mathsf{opt}}, \ldots, c_n^{\mathsf{opt}} = \operatorname*{argmax}_{c_1,\ldots,c_n} \frac{1}{2m}\sum_{i,j}\left(W_{ij} - \frac{D_{ii}D_{jj}}{2m}\right)\delta(c_i, c_j).$$

The authors impose two clusters, and set an edge weighting threshold equal to 0.9, meaning that all the edges with lower weights are set to 0. Figure 5.3 shows an example of the results obtained using the modularity optimization algorithm.

Pixel-level masks

Once an image is classified as forged and a community partition has been performed, labels are assigned to the nodes. In our setting, nodes are patches, which are usually taken with overlap. Therefore, it is necessary to aggregate this information to produce pixel-wise masks. To do so, Mayer and Stamm first construct a pixel-map. For the case in which the number of clusters is equal to 2, as it is in our setting, we can assume that there are only two labels: 0 and 1. The pixel-map can be then expressed as

$$P(x, y) = \frac{1}{\#\{R : (x, y) \in R\}} \sum_{R:(x,y)\in R} c(R), \tag{5.10}$$



Forged image

Forgery mask 1

Forgery mask 2

Extracted patches

Weights matrix

Detected communities

Figure 5.3: Examples of weight matrices and graph partitions on a forged image from the MISD Dataset [109], having two spliced regions. The community detection algorithm used for these examples is the modularity optimization, with patches of size 128×128 and 50% of overlap. The edge weight threshold used is equal to 0.9. We observe that the community partition found by the algorithm points to the forgery masks (in red).

where (x, y) is a pixel, R is a patch and $c(R)$ is the predicted label for the patch. The factor $\frac{1}{\#\{R:(x,y)\in R\}}$ normalizes the pixel-map to take into account the fact that pixels near the borders are covered by fewer patches.

This pixel-map is further smoothed using a 32×32 Gaussian kernel and compared to a threshold in order to produce a binary output. The threshold used by the authors is equal to 0.5. Finally, the smallest region is marked as forged and the largest one as pristine.

5.4 Demo

The online demo takes as input a suspicious image. The goal is to firstly classify the image in forged or pristine and, in the former case, to localize the forgery. The user is required to select the size of the patches, being 128×128 and 256×256 the two available options. The user can also decide the overlap with which these patches will be taken (50% or 75% of the patch size). Finally, the user can choose whether to use spectral clustering 5.3 or modularity optimization 5.3.

The demo prints the classification in tampered or non-tampered, according to the eigengap if spectral clustering is the chosen clustering algorithm, or according to the optimal modularity in case modularity optimization is chosen. The thresholds used to decide between tampered and non-tampered are those used by Mayer and Stamm, as explained in Section 5.3. If the image is classified as non-forged, a black mask is shown as output. On the other hand, if the image is classified as tampered, forgery localization is performed according to the chosen clustering technique, and the output mask shows the obtained results.

5.5 Experiments

5.5.1 Forged images

We first test the proposed approach on forged images. In these cases, the method should manage to identify the image as forged, and then perform forgery localization. Figure 5.4 shows the results obtained on some images from [97], for both clustering techniques and both patch sizes. In all these experiments, the patch overlap was set to 75%.

We observe that in most cases the method is able to detect the images as forged, except for one of the examples when using the spectral clustering technique and blocks of size 128×128. Interestingly, the *eigengap* seems to be very sensitive to the patch size, with significant lower values for larger patches. On the other hand, the optimal modularity seems to be more stable.

Regarding forgery localization, both techniques show significant agreement between the ground truth mask and the estimated mask. However, modularity optimization shows some falsely detected regions that are not present when using spectral clustering.

5.5.2 Authentic images

In these experiments we assess the performance of the studied approach when the input image is not tampered. Figure 5.5 shows examples of the obtained results for several authentic images from [97].

The method is able to correctly identify the images as non-tampered in most cases. However, spectral clustering, when patches are of size 256×256, delivers some false detections, as well as modularity optimization when used with 128×128 patches. These false detections seem to correspond to image regions having a distinctive texture. However, not all textured zones generate false alarms, as can be seen in the second image were none of the clustering techniques have false detections, despite the fact that the image has textured and flat zones.

An exhaustive quantitative evaluation of the described method is presented in Chapter 6.

5.6 Conclusions

In this Chapter we introduced and analyzed the method introduced in the article "Exposing Fake Images with Forensic Similarity Graphs" by O. Mayer and M. C. Stamm. In addition, we provide an online demo at: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=432. As an alternative to the methods presented in the previous chapter, the one studied here attempts to learn camera related features. Such an approach seems effective according to the qualitative experiments shown in Section **??**. Still, the method delivers some false detections and, in addition, the *eigengap* seems to be very sensitive to the patch size, with significant lower values for larger patches.

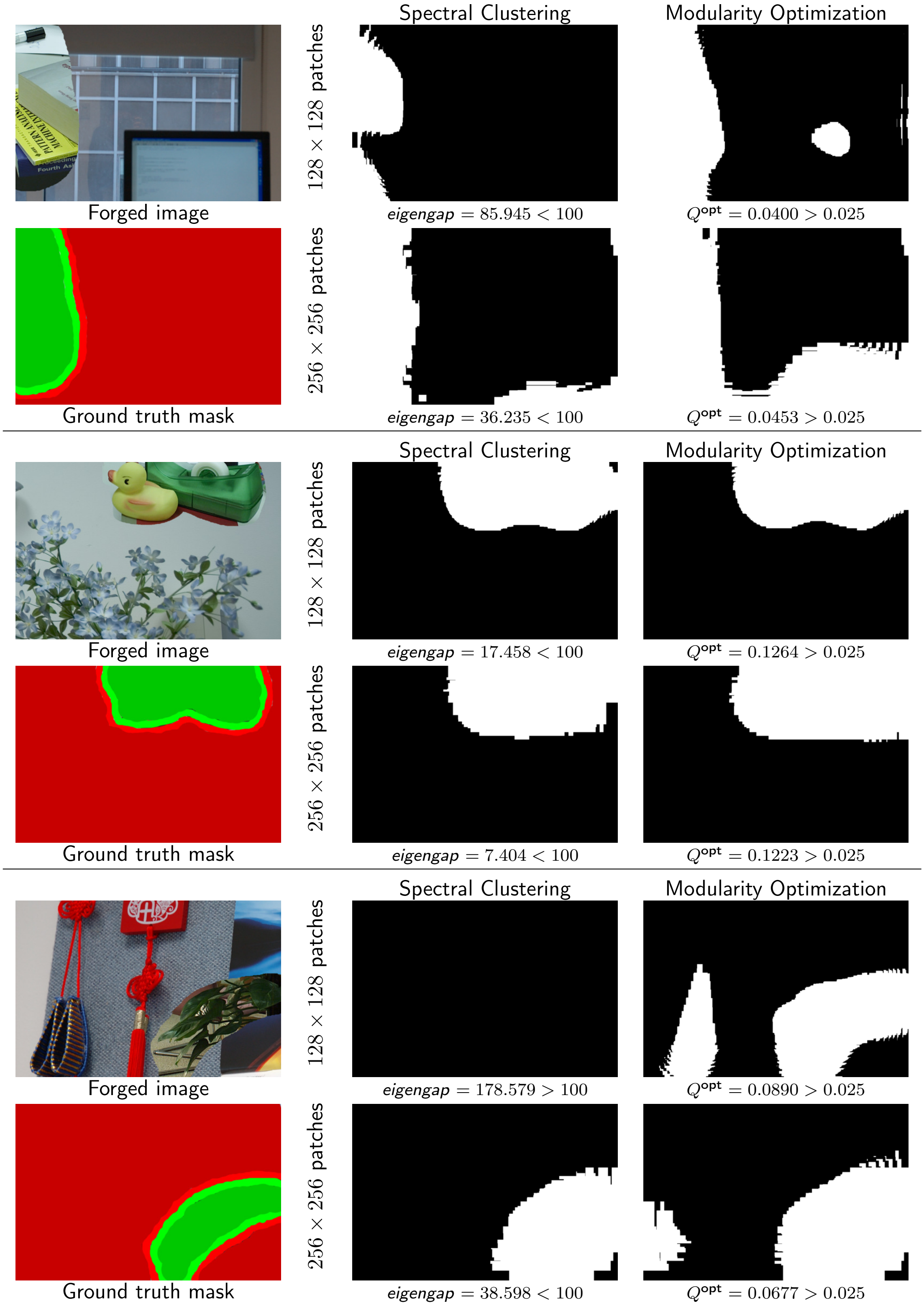

Spectral Clustering
Modularity Optimization
128 × 128 patches
Forged image
eigengap $= 85.945 < 100$
$Q^{\text{opt}} = 0.0400 > 0.025$
256 × 256 patches
Ground truth mask
eigengap $= 36.235 < 100$
$Q^{\text{opt}} = 0.0453 > 0.025$
Spectral Clustering
Modularity Optimization
128 × 128 patches
Forged image
eigengap $= 17.458 < 100$
$Q^{\text{opt}} = 0.1264 > 0.025$
256 × 256 patches
Ground truth mask
eigengap $= 7.404 < 100$
$Q^{\text{opt}} = 0.1223 > 0.025$
Spectral Clustering
Modularity Optimization
128 × 128 patches
Forged image
eigengap $= 178.579 > 100$
$Q^{\text{opt}} = 0.0890 > 0.025$
256 × 256 patches
Ground truth mask
eigengap $= 38.598 < 100$
$Q^{\text{opt}} = 0.0677 > 0.025$

Figure 5.4: Results obtained using the demo for some spliced images from the Columbia dataset [97]. The overlap is set to 75% of the patch size in all cases.

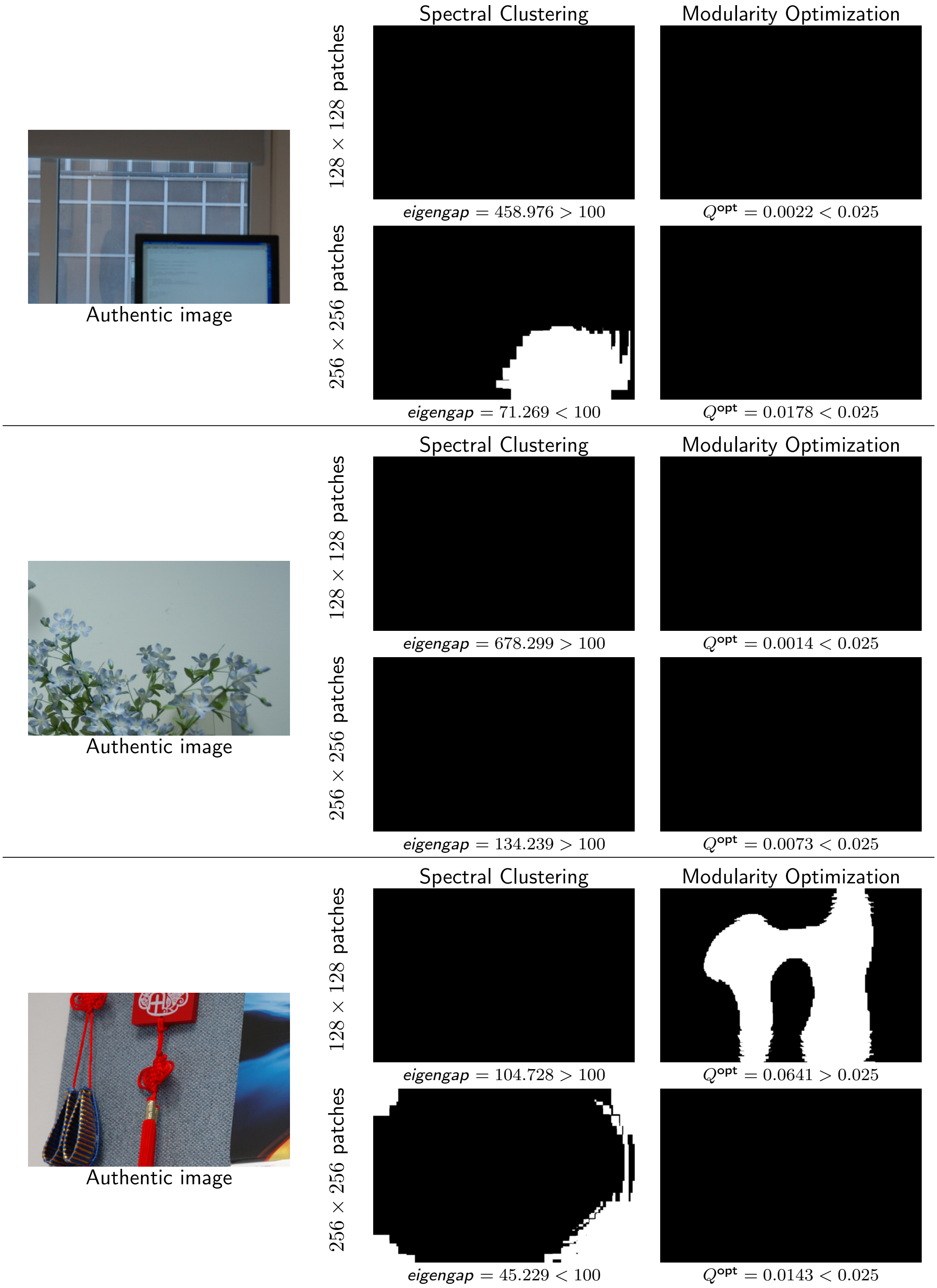


Figure 5.5: Results obtained using the demo for some authentic images from the Columbia dataset [97]. The overlap is set to 75% of the patch size in all cases.

Esta página ha sido intencionalmente dejada en blanco.

Chapter 6

Non-Semantic Evaluation of Image Forensics Tools

In this chapter we propose a new method to evaluate image forensics tools, that characterizes what image cues are being used by each detector. Our method enables effortless creation of an arbitrarily large dataset of carefully tampered images in which controlled detection cues are present. Starting with raw images, we alter aspects of the image formation pipeline (see Chapter 2) inside a mask, while leaving the rest of the image intact. This does not change the image's interpretation; we thus call such alterations "non-semantic", as they yield no semantic inconsistencies. This method avoids the painful and often biased creation of convincing semantics. All aspects of image formation (noise, CFA, compression pattern and quality, etc.) can vary independently in both the authentic and tampered parts of the image. Alteration of a specific cue enables precise evaluation of the many forgery detectors that rely on this cue, and of the sensitivity of more generic forensic tools to each specific trace of forgery, and can be used to guide the combination of different methods. Based on this methodology, we create a database and conduct an evaluation of the main state-of-the-art image forensics tools, where we characterize the performance of each method with respect to each detection cue. This work is published as *Non-Semantic Evaluation of Image Forensics Tools: Methodology and Database* in the WACV conference [13].

6.1 Introduction

Digital images play an extensive role in our lives and forgeries are present everywhere [64]. Creating visually realistic image alterations is easy.

Yet each modification of the image imprints traces onto it, that are cues of the tampering. Some forgery detection tools aims at detecting a specific trace in a suspicious image by finding local inconsistencies, while other methods, usually learning-based, are more generic. Semantic analysis of an image can provide hints, but the rigorous proof of a forgery should not be solely semantic. The situation is akin to the dilemma arising from the observations of Galileo, which contradicted the accepted knowledge of his time. In the words of Bertolt Brecht [24]:

> Galileo: How would it be if your Highness were now to observe these impossible as well as unnecessary stars through this telescope?
> The Mathematician: One might be tempted to reply that your telescope, showing something which cannot exist, may not be a very reliable telescope, eh?

The telescope could have been unreliable, indeed, and a scientific inquiry on the instrument could have been justified. However, concluding, as the Mathematician does, that the telescope was unreliable *just* based on the contents of the observations is not prudent. Similarly, the proof of a forgery must be based on image traces, not on semantic arguments, because the semantics of an image are usually the purpose and not the means of a forgery.

Image forensics algorithms are mainly evaluated by their performance in benchmark challenges. This practice has several limitations: in many cases, the same database is split into training and evaluation data. As a consequence, algorithms are trained and evaluated on images that have gone through similar image processing pipelines, forgery algorithms and anti-forensic tools. Hence, there is no guarantee that such learning-based methods will work in the wild, where those parameters vary much more. Regardless of the variety of the training set, the question arises of whether the forgeries are being detected by trained detectors for semantic reasons, or because of local inconsistencies in the image.

With these considerations in mind, we propose a methodology and a database to evaluate image forensic tools on images where authentic and forged regions only differ in the traces left behind by the image processing pipeline. Using this methodology, we create the *Trace database* by adding various forgery traces to raw images from the Raise [52] dataset, as shown in Figure 6.1. This procedure avoids the difficulties of producing convincing and unbiased semantic forgeries, which often requires manual work. We create several datasets, each of which corresponding to a specific pipeline inconsistency, such as a different noise level or compression pattern. This gives us insight into the sensitivity of forensic tools to specific traces, and thus highlights the complementarity of different methods.

Our contribution is twofold:

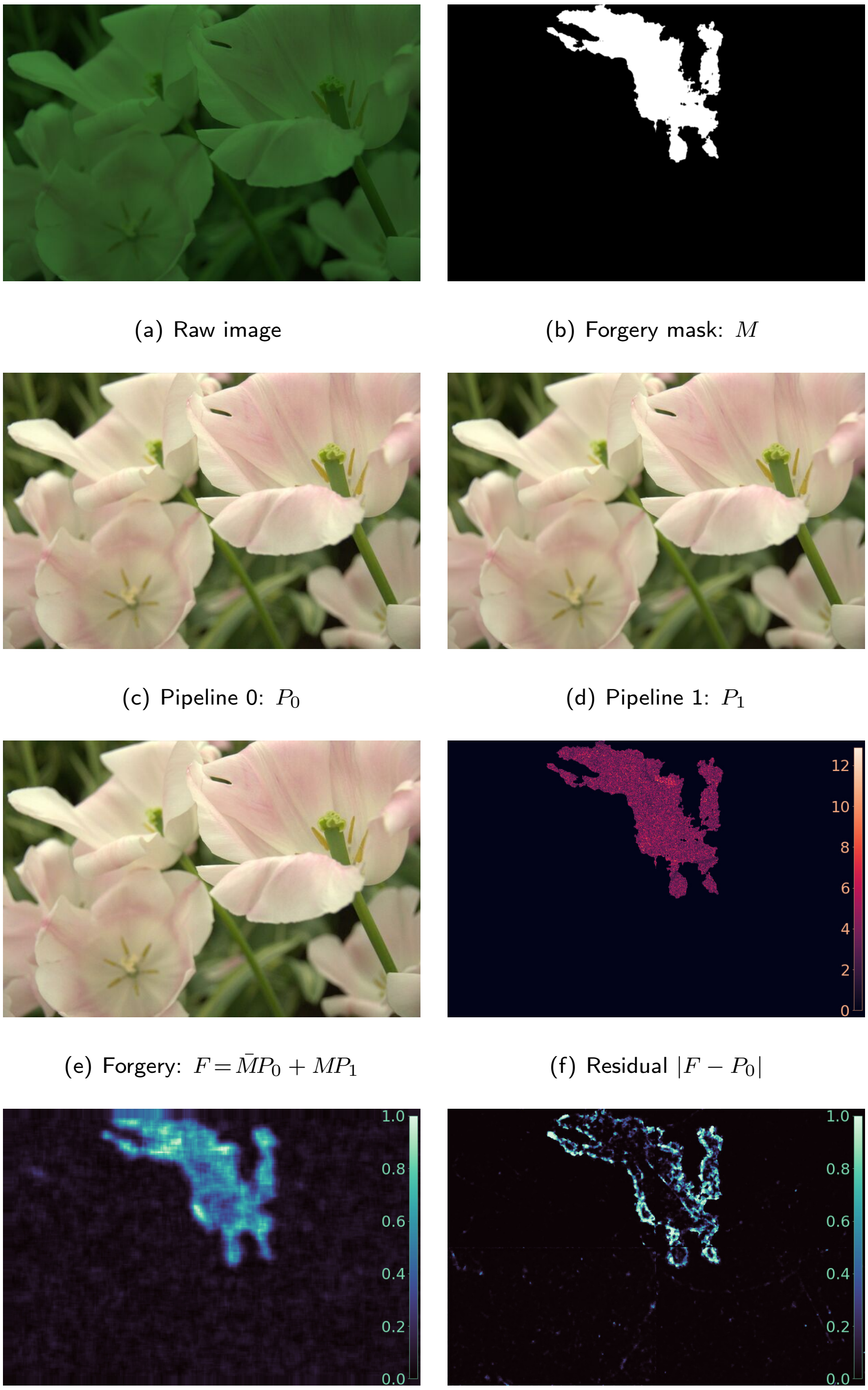
- we create a database of “fake” images with controlled inconsistencies in their formation pipeline,
- using this database, we conduct an evaluation of existing forensic tools.

6.2 Related Works

There is a large literature on image forensics, starting from the seminal work of Farid [64]. Some methods focus on the detection of a specific tampering attack such as copy-move or splicing, but the most classic forgery detection methods aim at detecting local perturbations of the traces left in the image by the processing chain. Such local disruptions hint at a local forgery. To do so, these methods strive to suppress image content and highlight intrinsic artefacts left by demosaicking, JPEG encoding, etc. [180]. These forgery detection methods can therefore be grouped by their specifically-targeted traces, which we now briefly review.

Noise-level-based methods analyse the noise model of images (see Section 6.3) to find regions with a different amount of noise, that could result from tampering. Mahdian and Saic [155] perform local wavelet-based noise level estimation using a median absolute deviation estimator. Lyu et al. [154] relies on the kurtosis concentration phenomenon. More recently, Noisesniffer (from Chapter 4) defines a background stochastic model enabling the detection of local and statistically-significant noise anomalies. These methods can potentially detect a relatively wide variety of forgeries, as each can alter the noise level.

Detecting the specific image **demosaicing** algorithm (see Section 6.3) has not been attempted since the 2005 pioneer paper by Popescu and Farid [181], conceived at a time where those algorithms were simpler and easier to distinguish, although some generic noise-pattern



(a) Raw image

(b) Forgery mask: M

(c) Pipeline 0: P_0

(d) Pipeline 1: P_1

(e) Forgery: $F = \bar{M}P_0 + MP_1$

(f) Residual $|F - P_0|$

(g) Noiseprint result

(h) Mantranet result

Figure 6.1: Different image formation pipelines are applied to the same RAW image to obtain two images, that are combined to obtain a forged image. The authentic and forged regions present different camera pipeline traces, but are otherwise perfectly coherent. The last row shows the result of two forensic tools on this image.

analysis method can distinguish different algorithms given large enough regions [48]. However, detecting the mosaic pattern has received more extensive coverage. Choi et al. [35] used the fact that sampled pixels were more likely to take extremal values, while Shin et al. [192] noticed that they had a higher variance. Bammey et al. [10] combined the translation invariance of convolutional neural networks with the periodicity of the mosaic pattern to train a self-supervised network into implicitly detecting demosaicing artefacts. Because demosaicing artefacts lie in the high frequencies, they are lost under a strong JPEG compression or when the image has been downsampled. As such, they are usually best used on high-quality images.

JPEG compression leaves blocking effects and quantization of the DCT coefficient of each block. JPEG forensic tools can thus be divided into two categories. BAG [138] and CAGI [100] analyse blocking artefacts, while other methods analyse the DCT coefficients. More precisely, CDA [143] and I-CDA [20] are based on the AC coefficient distributions, while FDF-A [4] is based on the first digit distribution of AC coefficients. Zero [176] counts the number of null DCT coefficients in all blocks and deduces the grid origin. These methods can only work when the forgery was done after a first JPEG compression. And when this is the case, they usually yield very good results.

In the past few years, **multi-purpose** tools were proposed to detect inconsistencies from multiple traces simultaneously. Similarity Graphs (from Chapter 5) trains a siamese network on image patches to predict the similarity in their forensic traces. The similarity score is then used to construct a graph representation of the image, where community detection is performed to spot forgeries. Splicebuster [49] uses the co-occurences of noise residuals as local features revealing tampered image regions. Noiseprint [48] extends on Splicebuster and uses a Siamese network trained on authentic images to extract the noise residual of an image, which is then analysed for inconsistencies. ManTraNet [221] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localise forgeries in the image. Finally, Self-consistency [99] analysis also uses a Siamese network with the goal of detecting whether two patches have been processed with the same pipeline. They make use of N-Cuts segmentation [104] to automatically cluster and detect relevant traces of forgeries. With these methods, exhaustiveness is theoretically possible. However, results are not self-explanatory and those method's decisions are harder to justify. Furthermore, learning-based methods can be limited by the training data, and may fail to generalize well in uncontrolled scenarios.

There is also considerable literature proposing datasets for the evaluation of forensic tools. An early example is the Columbia Dataset [173], which only contains spliced 128×128 grayscale blocks for which no masks are provided. New benchmarks were proposed in 2009 with CASIA V1.0 and V2.0 [58]. These datasets included splicing and copy-move attacks, with a total of 8000 pristine images and 6000 tampered images. Post-processing was introduced as a counter-forensics technique. MICC F220 and F2000 datasets [3] as well as the IMD dataset [37] provide further benchmarks for copy-move detection. These datasets were constructed in an automatic way. While the first two randomly select the region of the image to be copy-pasted, IMD dataset performed snippets extraction. Other datasets adressing copy-move forgeries with post-processing counter attacks are also available [209, 215].

Image forgery-detection challenges are another source of benchmark datasets. The National Institute of Standards and Technology (NIST) organizes, since 2017, an annual challenge for which different datasets are released [89]. It includes automatically and manually generated forgeries of considerable variety, and can thus be useful to evaluate image forgery detection in uncontrolled scenarios.

Some datasets aim at performing forgeries imperceptible to the naked eye. A good example is the Korus dataset [121, 122] which contains 220 pristine images and 220 handmade tampered images targeting object removal or insertion.

The recent DEFACTO dataset [156] is constructed on the MSCOCO dataset [140] and includes a wide range of forgeries such as copy-move, splicing, inpainting and morphing. Semantically meaningful forgeries are generated automatically but with several biases such as copy-pasting objects in the same axis or only performing splicing with simple objects.

Most recent forgery-detection datasets start from pristine images and perform several sorts of forgeries on them [238]. Since the creation of early datasets [58, 97, 173], the number of tampering techniques has increased to include new ones such as colorization [26], inpainting [26, 156] and morphing [156, 239]. Post-processing and counter-forensic techniques have been increasingly used to produce visually imperceptible forgeries; but such approaches may also introduce detectable traces.

Efforts have also been made to automatically obtain large datasets. Yet, the resulting forged images are either semantically incorrect [3, 37] or biased [156]. Both scenarios pose problems for training neural networks, which risk overfitting on the forgeries' methods and semantic content.

The variety of forgery methods makes the evaluation of forensic tools difficult to interpret, as the performance depends on the suitability of the detection tool for the specific forgery method. In quantitative experiments, using multiple datasets, and especially datasets with varied forgeries, helps assess the quality of a forensic tool. However, those results also become harder to interpret. On the other hand, while results using the proposed database will not be reflective of uncontrolled scenarios, they help precisely identify which traces a forensic tool can and cannot detect.

6.3 Image formation pipeline

In this section we summarise the image formation pipeline, which was already introduced in Chapter 2. Figure 6.2 summarises the main steps [54] and shows how the noise curves change at its different steps.

Raw image acquisition

The value at each pixel can be modelled as a Poisson random variable [67]. Noise variance at this step thus follows an affine relation $\sigma^2 = A + Bu$ where u is the intensity of the ideal noiseless image and A and B are constants (see Figure 6.2). Furthermore, given the nature of the noise sources at this step, noise can be accurately modelled as uncorrelated, meaning that noise at one pixel is not related with the noise at any other pixel.

Demosaicing

Most digital cameras are equipped with a single sensor array. In order to obtain a colour image, a colour filter array (CFA) is placed in front of the sensor to split incident light components according to their wavelength. The raw image obtained from the sensor therefore is a mosaic containing a single colour component per pixel: red, green, or blue. Demosaicing methods interpolate the missing colours at each pixel to reconstruct a full colour image. After demosaicing (Figure 6.2), each channel has a different noise curve, and noise becomes spatially correlated.

Colour Correction

In order to obtain a faithful representation of the colours as perceived by the observer, white balance adjusts colour intensities in such a way that achromatic objects from the real scene are rendered as such [149]. This is done by scaling each channel separately, thus also scaling differently the noise level of each channel. Given that the relationship between stimulus and human perception is logarithmic [65], cameras then apply a power law function

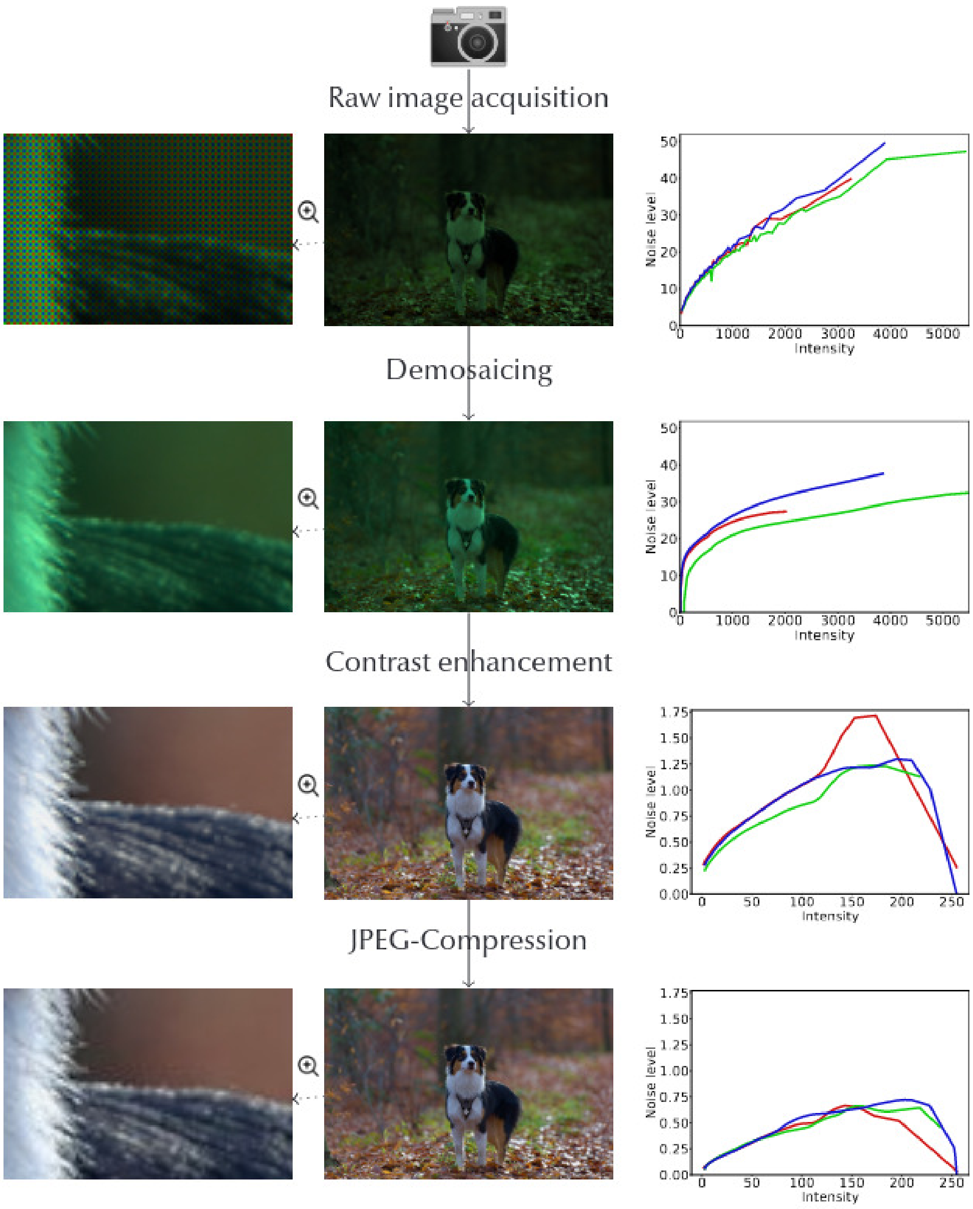


Figure 6.2: Evolution of the noise curves when passing through the successive steps of a (simplified) image processing pipeline.

to the intensity of each channel. After this step, known as gamma correction, the noise level is no longer monotonously increasing with the intensity.

JPEG compression The JPEG image standard is the most popular lossy compression scheme for photographic images [211]. The image goes through a colour space transformation and each channel is partitioned into non-overlapping 8×8-pixel blocks. The type-II discrete cosine transform (DCT) is applied to each of these blocks. The resulting coefficients are quantized according to a table and the coefficients are then compressed without additionnal loss. Due to the cancellation of high-frequency coefficients, the noise is reduced after compression.

6.4 The Proposed Methodology

We created a database of “forged” images which leaves the semantics of the images intact. The overall idea of our method is to take a raw image, process it with two different pipelines, and merge the two processed images as follows: the first image is used for the authentic region and the second image for the “forged” area determined by a mask, as can be seen in Figure 6.1. As a base we use the RAISE-1k dataset [52], which contains one thousand pristine raw images of varied categories, taken from three different cameras. We note that the variety of source cameras is not important to our database, as we erase the previous camera traces by downsampling the image, then resimulate the whole image processing pipeline ourselves, as explained below. Furthermore, our open source generation code can be applied on any other source of images, to automatically generate arbitrarily large quantities of “forged” images.

Methodology for the creation of the database A raw image already contains noise, furthermore its pixels are all sampled in the same CFA pattern. In order to reduce the noise and eliminate the CFA pattern, we start by downsampling each image by a factor 2. This enables us to choose the amount of noise to be added, and to mosaic the image in any of the four possible patterns. Once the image has been downsampled, we process the image with two different pipelines. The two images are then merged as explained above.

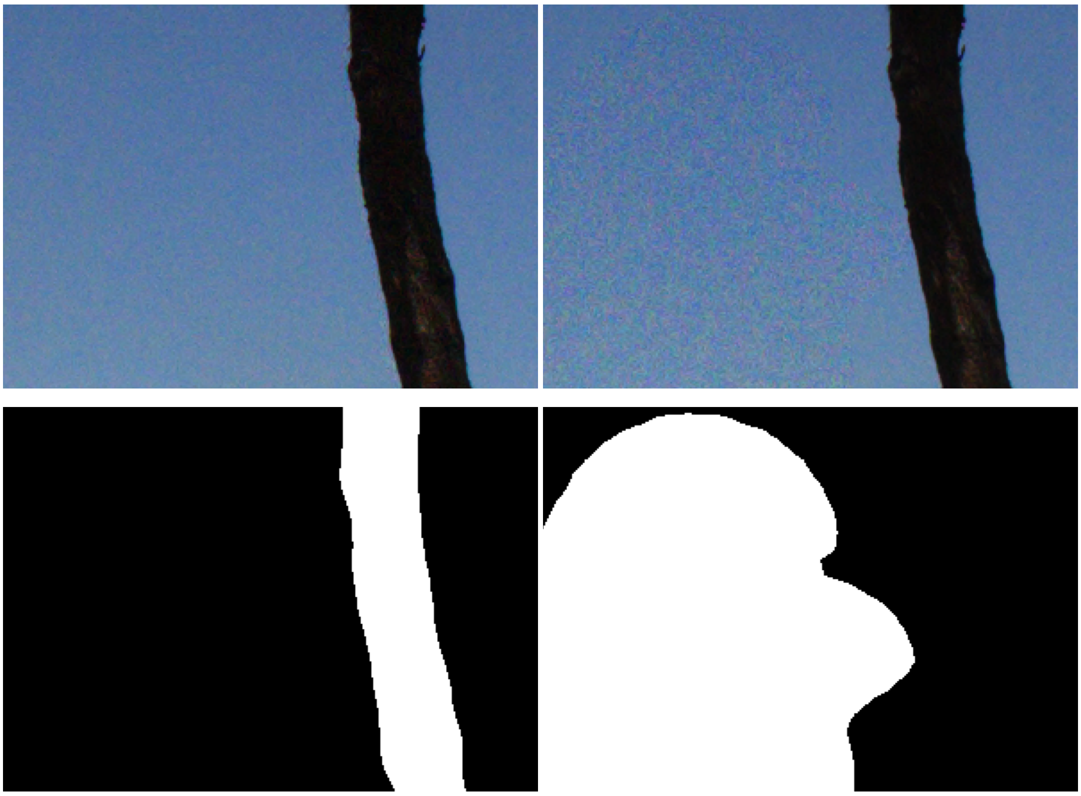


Figure 6.3: Details of the same image with forgeries made using the two masks. On the left, the endomask coincides with the image’s structure, here a tree. The forgery is less conspicuous than on the right where the exomask is in the sky, where the borders do not coincide with the images’ content.

Forgery masks For each image we construct two different kinds of masks, which we shall call *endomasks* and *exomasks*. Since inconsistencies in the image processing pipeline are usually most visible at the border of the forgery, *endomasks* are obtained as regions of a segmentation of the image. To do this, we segment the original images with EncNet [234]. For each image, we take a pixel at random, and select the image region it belongs to. We accept the mask if its size is less than half the image's, otherwise we pick another pixel until we find a suitable mask. This ensures that each image has only one forgery, whose size is at most half the image's. Using such endogenous masks or *endomasks* corresponding to a region of the segmented image ensures almost invisible forgeries. Indeed their borders are natural image borders, as shown in Figure 6.3.

The *exomasks* are instead unrelated to the image's content. To determine them, we start by pairing the images of the dataset according to their endomasks' sizes. Then, the endomask of each image is used as the exogenous mask, or *exomask*, of its paired image. Using a mask from another image ensures that the mask is not linked to the image's semantic. The chosen pairing enables comparisons separately on each image, as the size of the masks is similar. See Figure 6.4 for examples of endo- and exomasks.

Multiple datasets One of our goals is to determine which inconsistencies each forensic tool is sensitive to. Changes in the image processing pipeline, done at different steps of the chain, lead to different inconsistencies (see Section 6.3). In consequence, we created five specific datasets, each of which features a specific change in the image processing pipeline. For each image, we started by randomly choosing the three parameters that are used for this image across all datasets:

- The mosaic pattern, chosen among the four possible offsets of the camera's Bayer pattern.
- The demosaicing algorithm, chosen randomly among those available in the LibRaw library [139].
- The gamma-correction power.

The gamma correction is the same for both regions of the image, and the mosaic pattern is the same except for the CFA Grid, CFA Algorithm and Hybrid datasets. For each image, both the endo- and exomasks, constructed as explained above, are the same across all datasets.

Raw Noise Level dataset In this dataset we add random noise to each raw image before processing it. As pointed out in Section 6.3, noise variance in raw images follows a linear relation given by $\sigma^2 = A + Bu$, where A and B are constants and u is the noiseless image. We start by randomly selecting two different pairs of constants (A_0, B_0) and (A_1, B_1), in a range that ensures the resulting images look natural. Both images are then processed with the same pipeline. This dataset mimics the inconsistencies in noise models that could be found in spliced images.

CFA Grid dataset In this dataset we only change the mosaic pattern of the forged image inside the mask. Thus, the original image and the forged one would be identical if not for their mosaic grid origins. This kind of trace may appear (with probability $\frac{3}{4}$) when the forgery was an internal copy-move. Indeed, even if the forged region has a similar signature, there is no reason the mosaic grid of the forged region should be the same as in the authentic region unless the copy-move translation is a multiple of 2 in both directions.

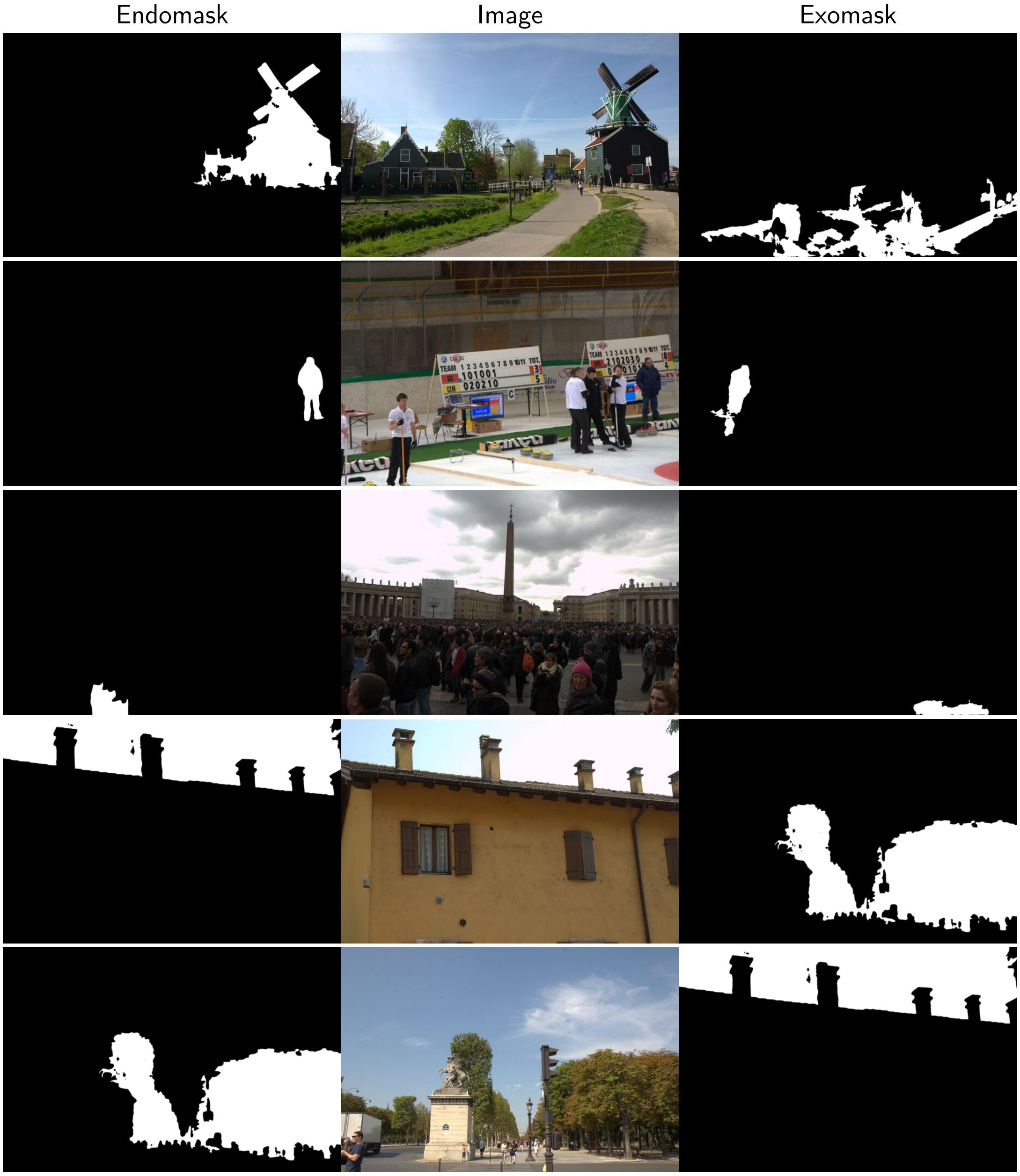


Figure 6.4: For each image, we use an *endomask* (left) taken from the image's segmentation, and an *exomask* (right) taken from another image and thus decorrelated from the image's contents. The last two images were paired during mask creation, thus the endomask of each becomes the exomask of the other.

CFA Algorithm dataset In this dataset, the two processing pipelines use different demosaicing algorithms. The demosaicing pattern is chosen independently for each pipeline. Thus there is a $\frac{1}{4}$ chance that they are aligned.

A new mosaic pattern is also randomly chosen, thus having a $\frac{3}{4}$ chance of being different from the one of the main image. This dataset represents the change in the mosaic that would occur from splicing, as two different images most likely do not share the same demosaicing algorithms, and the alignment of their patterns after splicing is random.

JPEG Grid dataset In this dataset we only change the compression grid origin. Similarly to the CFA Grid dataset, if the forgery is an internal copy-move, the JPEG grid of the forged region is different from the grid in the authentic region, with probability $\frac{63}{64}$. The JPEG compression quality used in both pipelines is then chosen randomly, keeping the values in a range that is typical of most compressed images and challenging enough for JPEG-based algorithms.

JPEG Quality dataset In this dataset, both the authentic and forged regions are processed with the same pipeline, except for the JPEG compression which is done in the two regions with different quality factors, again chosen uniformly between 75 and 100. Like with the CFA Algorithm dataset or the JPEG grid data, a new JPEG grid pattern is also randomly chosen, which has a $\frac{63}{64}$ chance of being different from the main region's grid. This dataset simulates the effect of the splicing of an image onto another, both images being compressed at different quality factors.

The hybrid dataset One could argue that although generic learning-based forensics tools may not be able to point out a single inconsistency in an image, they might be best suited to find multiple inconsistencies stacked together. Clearly, a splicing may introduce joint inconsistencies in noise level, JPEG encoding and demosaicing; while a direct copy-move can introduce alterations in the JPEG and CFA grids. To investigate such possibilities, in addition to the five specific datasets described above, we created a sixth, hybrid dataset. In this dataset, forgeries combine noise, demosaicing and/or JPEG compression traces. At least two of those traces are altered in each images.

To create this dataset, we adopt the following procedure for each image:

1. We randomly choose whether to modify two or three steps of the pipeline (added noise, demosaicing grid/method, JPEG grid/quality). If we only change two, we select which steps to change.

2. For JPEG and CFA modifications, we select whether we only change the CFA and JPEG grids, or if we change the demosaicing methods, the JPEG quality factor and potentially the CFA and JPEG grids. The decision is made jointly for JPEG and CFA, as the CFA and JPEG Grid datasets mimic artefacts commonly found in internal copy-move forgeries, whereas the CFA Algorithm and JPEG Quality datasets represent inconsistencies more typical of splicing.

3. Finally, for each different change, we select its parameters in the same way as for the specific datasets.

In short, each image has a minimum of two parameters that vary between the two parts. At the maximum, all studied parameters in this chapter (raw noise level, demosaicing pattern and algorithm, JPEG grid and quality factor) can vary between the two regions.

6.5 Experiments

6.5.1 Evaluated methods

We used the constructed database to conduct an evaluation of image forensics tools. We tested both classic and SOTA forgery detection methods pertaining to different traces: noise-level-based detection methods Noisesniffer (from Chapter 4), Lyu [154, 230] and Mahdian [155, 230]; CFA-grid detection methods Bammey [10], Shin [192] and Choi [11, 35]; JPEG-based methods Zero [176], CAGI [100, 230], FDF-A [4, 230], I-CDA [20, 230], CDA [143, 230] and BAG [138, 230], as well as generic methods Similarity Graphs (from Chapter 5), Splicebuster [49], Noiseprint [48], ManTraNet [221] and Self-Consistency [99].

6.5.2 Evaluation Metrics

We evaluated the results of these methods using the Matthews correlation coefficient (MCC) [161]. This metric varies from -1 for a detection that is complementary to the ground truth, to 1 for a perfect detection. A score of 0 represents an uninformative result and is the expected performance of a random classifier. The MCC is more representative than the F1 and IoU scores [33, 34], partly as it is less dependant on the proportion of positives in the ground truth, which is especially important given the large variety of forgery mask sizes in the database.

The MCC was computed for each image, and then averaged over each dataset. As most surveyed methods do not provide a binary output but a continuous heatmap, we weighted the confusion matrix using the heatmap.

6.5.3 Results

The complete results are given in Table 6.1. Visualization of the detection by several methods on one image across all datasets can be seen in Figure 6.5. In the CFA and JPEG datasets, state-of-the-art methods that focus on those specific traces, such as Bammey [10] for CFA and ZERO [176] for JPEG, perform much better than generic tools. This is partly expected, as those methods aim to detect exactly this specific trace. Still, some generic methods such as Similarity Graphs (from Chapter 5) achieve a competitive performance.

This observation is more nuanced in the Noise Level dataset where, depending on the type of mask considered, Noisesniffer (from Chapter 4) and Self-Consistency [99] achieve the best results. Indeed, exomasks cover a wider range of intensities enabling a better comparison between noise models, which is exploited by Noisesniffer. Also, half of the forgeries present in this database are undetectable for this method since it is only able to detect forgeries having lower noise levels.

On the hybrid dataset, the scores of the specific methods are lower than on the specific datasets. For the JPEG-based methods, this is explained by the fact that one sixth of this dataset does not feature JPEG compression traces. For the CFA and Lyu and Mahdian noise-based methods, this is made worse by the fact that JPEG compression alters the previous noise and demosaicing artefacts, as shown in Figure 6.2. In particular, CFA-based methods are notoriously weak on JPEG images, since JPEG compression removes the high frequencies, in which mosaic artefacts lie. This can be seen in Figure 6.5, where the CFA-based method Bammey cannot make any prediction on the hybrid image, where the main and forged region were compressed with quality factors of 93 and 75, respectively.

While multi-purpose forensic methods can, to some extent, detect noise-level inconsistencies, in the demosaicing algorithm and in the JPEG quality, they are blind to shifts in both the JPEG and CFA grids. This is not entirely surprising; with the exception of Splicebuster, the

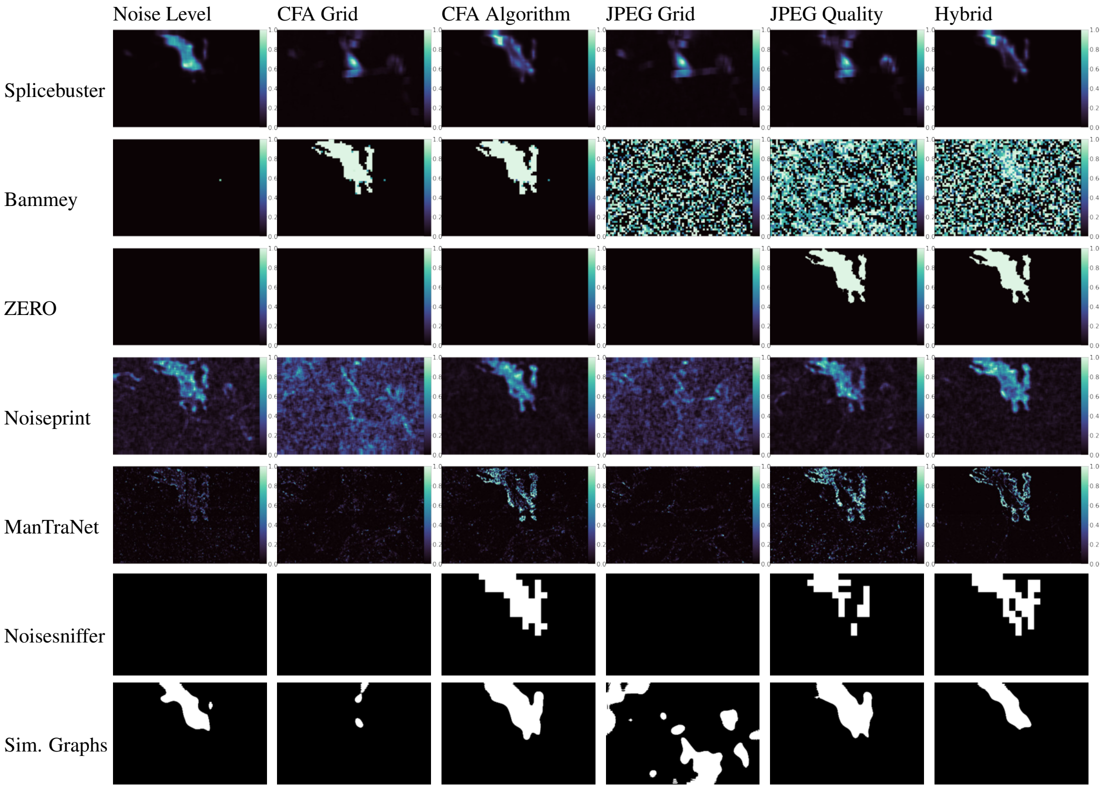


Figure 6.5: Visualization of the results of several methods for one image on all the datasets. Some methods, such as Noiseprint or Bammey, correctly detect the forgeries in the relevant images, but tend to make noise-like false detections in the images for which they cannot see the forgery. Automatically selecting the relevant detections of an algorithm would make it easier to use without needing interpretation. The image and mask can be seen in Figure 6.1.

tested generic tools are based on mostly-convolutional neural networks, which are invariant to translation. Although Noiseprint [48] adapts its training scheme to be able to detect shifts in periodic patterns, it cannot see the demosaicing and JPEG compression grids, although it is sensitive to JPEG quality inconsistencies and to some extent to demosaicing algorithm changes as well.

Most methods perform similarly on the endomask and exomask datasets. Two notable exceptions are Noisesniffer which underperforms on endomasks, and Self-Consistency, which works much better on endomasks. Both observations are easily explained: when using exogeneous masks, forgeries have more heterogeneous intensities. Since the method performs intra-bin comparisons, the wider range of intensities benefits the detection of the manipulated zone since more bins are involved. In contrast, Self-consistency's content-awareness is lost when segmenting forgeries with exomasks. Regardless of the dataset considered, the scores obtained by all of the methods have a high standard deviation with respect to their mean value. This suggests that, given a dataset, the scores in each individual image are not concentrated around the mean but rather spread on a large range of values. Hence, even for methods having low scores, some good detections are likely to happen.

6.6 Discussion

Most methods yield similar results on exo- and endomasks. While one kind is usually sufficient, comparing the results on both shows some methods are content-aware.

The goal of this evaluation was not to rank different methods, but to offer a rigorous insight on the capabilities of each. Knowing to which kind of inconsistencies forensic tools are sensitive helps understand and explain its detections in uncontrolled cases, and can help efforts to combine different methods. In that sense, the proposed database is complementary to more traditional databases.

Even though many methods can yield decent scores, the standard deviations of theses scores over all images of the same dataset is often very high. Even though algorithms perform well on many forgeries, they also often yield false positives that require interpretation to be distinguished from true detections, such as Figure 6.1. This is a critical point for many methods, as it makes them usable only to a trained eye.

6.7 Conclusion

Image forensics datasets are usually grouped according to forgery types (eg. splicing, inpainting, or copy-moves), and do not separate the semantic content from the actual traces left by the forgery. In this chapter, we proposed to remove the semantic value of forgeries and to focus only on the traces. We designed a methodology to automatically create image "forgeries" that leave no semantic traces, by introducing controlled changes in the image processing pipeline. We built datasets by focusing on noise-level inconsistencies, mosaic and JPEG artefacts, and conducted an evaluation of some image forensics tools using this dataset.

Although we focused on three kinds of changes in the forgeries, the same methodology can be applied to more traces, including multiple compression, or image manipulations such as resampling. In fact, we can address all forgeries where two different camera pipelines are involved. This includes copy-move, splicing and some methods of inpainting. Further work would incorporate other traces, such as those left by synthesis methods. Although not surveyed here, the same methodology can be applied to study robustness of detection under adverse events such as global JPEG compression, by passing the images through compression before analysis. Our images were not post-processed, except for inconsistencies linked to JPEG compression. This makes it easy to assess the robustness to any kind of post-processing.

Note that there are no authentic images in the dataset. Testing the frequency of false positives is for now complementary to the proposed methodology, but could be included in further work by comparing the response of forensic tools to the forged images and their authentic counterparts, otherwise-processed with the same pipeline.

Our method can transform automatically large sets of images into forged images with fully controlled tampering cues and no bias that might cause overfitting. Besides evaluation of existing image forensics tools, this methodology could also be used to train forgery detection methods, although care would be needed so as not to overfit if using the same methodology for both training and evaluation.

		Dataset					
		Noise Level	CFA Grid	CFA Algorithm	JPEG Grid	JPEG Quality	Hybrid
Noise-level-based	Noisesniffer (Chapter 4)	0.192 (0.376)	-0.008 (0.121)	0.057 (0.256)	-0.010 (0.118)	0.072 (0.278)	0.129 (0.326)
		0.135 (0.317)	-0.014 (0.139)	0.003 (0.213)	-0.013 (0.137)	0.026 (0.224)	0.078 (0.274)
	Lyu [154]	0.010 (0.090)	0.002 (0.093)	0.002 (0.094)	0.000 (0.089)	0.002 (0.091)	0.012 (0.097)
		0.007 (0.137)	0.010 (0.157)	0.009 (0.159)	0.007 (0.148)	0.013 (0.156)	0.018 (0.150)
	Mahdian [155]	0.046 (0.146)	0.005 (0.082)	0.039 (0.128)	0.005 (0.086)	0.036 (0.132)	0.055 (0.158)
		0.055 (0.171)	0.023 (0.159)	0.057 (0.183)	0.014 (0.146)	0.052 (0.180)	0.067 (0.191)
CFA-based	Bammey [10]	0.007 (0.084)	0.682 (0.329)	0.501 (0.427)	0.023 (0.095)	0.029 (0.091)	0.133 (0.288)
		0.021 (0.153)	0.665 (0.349)	0.491 (0.429)	0.018 (0.107)	0.020 (0.100)	0.128 (0.290)
	Shin [192]	0.007 (0.101)	0.104 (0.166)	0.085 (0.172)	-0.002 (0.042)	-0.001 (0.043)	0.015 (0.109)
		0.004 (0.123)	0.099 (0.171)	0.084 (0.179)	-0.005 (0.058)	-0.006 (0.059)	0.012 (0.114)
	Choi [11, 35]	0.026 (0.025)	0.603 (0.203)	0.420 (0.208)	0.001 (0.002)	-0.001 (0.003)	0.156 (0.114)
		0.030 (0.018)	0.575 (0.191)	0.385 (0.210)	-0.001 (0.002)	0.001 (0.001)	0.139 (0.116)
JPEG-based	Zero [176]	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.796 (0.349)	0.732 (0.413)	0.638 (0.451)
		0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.756 (0.387)	0.708 (0.421)	0.624 (0.453)
	CAGI [100]	0.004 (0.045)	0.000 (0.027)	0.002 (0.033)	0.038 (0.077)	0.044 (0.080)	0.031 (0.071)
		0.003 (0.052)	0.000 (0.042)	0.001 (0.044)	0.023 (0.077)	0.028 (0.082)	0.021 (0.073)
	FDF-A [4]	0.031 (0.139)	-0.004 (0.087)	-0.003 (0.085)	0.226 (0.242)	0.228 (0.249)	0.203 (0.244)
		0.014 (0.169)	-0.015 (0.139)	-0.017 (0.139)	0.216 (0.265)	0.216 (0.273)	0.187 (0.264)
	I-CDA [20]	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.416 (0.417)	0.422 (0.407)	0.381 (0.407)
		0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.423 (0.408)	0.414 (0.414)	0.385 (0.408)
	CDA [143]	-0.001 (0.034)	0.000 (0.055)	0.000 (0.052)	0.485 (0.339)	0.474 (0.344)	0.401 (0.360)
		-0.004 (0.068)	-0.003 (0.098)	-0.005 (0.097)	0.449 (0.351)	0.442 (0.350)	0.378 (0.354)
	BAG [138]	0.000 (0.015)	0.006 (0.078)	0.009 (0.079)	0.232 (0.461)	0.229 (0.458)	0.171 (0.430)
		0.002 (0.029)	0.025 (0.164)	0.026 (0.164)	0.227 (0.459)	0.223 (0.455)	0.161 (0.430)
Multi-purpose tools	Similarity Graphs (Chapter 5)	0.109 (0.326)	0.063 (0.260)	0.144 (0.350)	0.525 (0.416)	0.580 (0.399)	0.495 (0.434)
		0.102 (0.357)	0.074 (0.329)	0.135 (0.388)	0.526 (0.430)	0.577 (0.405)	0.523 (0.441)
	Noiseprint [48]	0.127 (0.200)	-0.001 (0.069)	0.066 (0.149)	0.013 (0.087)	0.178 (0.248)	0.153 (0.230)
		0.108 (0.232)	0.002 (0.114)	0.060 (0.179)	0.016 (0.140)	0.138 (0.279)	0.128 (0.261)
	ManTraNet [221]	0.049 (0.091)	0.000 (0.040)	0.074 (0.169)	0.004 (0.023)	0.095 (0.164)	0.112 (0.169)
		0.032 (0.099)	-0.004 (0.065)	0.053 (0.165)	0.000 (0.043)	0.086 (0.171)	0.107 (0.176)
	Self-Consistency [99]	0.082 (0.323)	0.028 (0.261)	0.036 (0.270)	0.011 (0.262)	0.078 (0.335)	0.138 (0.370)
		0.154 (0.429)	0.077 (0.393)	0.082 (0.403)	0.060 (0.386)	0.151 (0.440)	0.246 (0.425)
	Splicebuster [49]	0.099 (0.188)	0.003 (0.085)	0.075 (0.157)	0.005 (0.083)	0.084 (0.175)	0.101 (0.192)
		0.100 (0.217)	0.012 (0.157)	0.072 (0.202)	0.006 (0.135)	0.082 (0.220)	0.099 (0.215)

Table 6.1: Results of different state-of-the-art forensics tools on our six datasets, using the Matthews Correlation Coefficient (MCC), detailed in Section 6.5.2. The methods, on the left, are grouped by categories. As a baseline, a random classifier is expected to yield a score of 0. The mean of the MCC scores over each image of the dataset, as well as the standard deviation in parentheses, are shown for the exogenous mask and endogenous mask datasets. Grayed-out numbers represent results of methods on datasets that are irrelevant to said methods. The best two scores are underlined for each database.

Part II

Source camera identification based on noise characteristics

Chapter 7

Analysis of the Forensic Similarity Approach for Source Camera Model Comparison

As pointed out in Chapter 2, the traces left by the image processing pipeline can be used for several forensics tasks rather than image forgery detection and localization. In this chapter, we show that the forensic similarity approach (Chapter 5) can also be used for source camera model comparison. Indeed, such an approach aims at determining whether two image patches share the same forensic traces or not. Images acquired with devices from the same model are expected to exhibit the same similar forensic traces, while devices from different models are expected to produce different traces. In order to make the chapter self-contained, the forensic similarity approach is explained again in Section 7.2.

The work presented in this chapter is published as *Forensic Similarity for Source Camera Model Comparison* in IPOL [71] and an online demo is available at: hhttps://ipolcore.ipol.im/demo/clientApp/demo.html?id=424.

7.1 Introduction

Providing information about the camera with which an image was acquired can be crucial for different forensic applications. Indeed, it can provide clues to track pornographic content, to check for copyright infringement, and to verify the consistency of a database. There are different approaches that aim at describing the source device of a given image. Some of them try to identify the particular device with which the image was taken [36, 151], while others focus on identifying the brand or model of the source camera [205, 206**?**].

Classic methods tackle this problem by searching for device traces. These traces include sensor pattern noise [151], lens distortions [36], demosaicing artefacts, white balance traces [55] and compression. Some of these features, such as the PRNU pattern [151] or radial distortion [36], are device-specific and can lead to accurate device identification. In particular, the use of PRNU traces for source camera certification is explored in Chapter 8. Other features are shared by different devices from the same model or brand, and can therefore provide information about the device model rather than identifying a particular source camera [207].

In this chapter we explore how the forensic similarity approach introduced in Chapter 5 can be applied for source camera model comparison. This approach aims at determining whether two image patches share the same forensic traces or not. Forensic traces are signals embedded in the image during the image formation process. Indeed, from the moment the light hits the camera sensors until the final digital file is delivered, the image undergoes several operations such as demosaicing, denoising, gamma correction, white balance and compression. Each of these operations leaves specific artifacts in the final image. Images acquired with devices from the same model are expected to exhibit the same similar forensic traces, while devices from different models are expected to produce different traces.

7.2 Method

7.2.1 Problem formulation

The problem can be stated as follows: Given two image patches, we want to assign a score of 0 to the pair of patches if they have different forensic traces, and a score of 1 if they share the same forensic traces. That is, we search for a map $C : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ is the space of all image patches, such that

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } X_1, X_2 \text{ have different forensic traces,} \\ 1 & \text{if } X_1, X_2 \text{ have the same forensic traces.} \end{cases}$$

This problem can be tackled in three steps. First, a suitable set of N features capturing the forensic information is extracted from each patch, by means of a feature extractor $f : \mathcal{X} \to \mathbb{R}^N$. The resulting feature vectors are then compared based on a similarity function $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$. Finally, the similarity measure is compared to a threshold τ so as to obtain a binary output. The map C can be then written as

$$C(X_1, X_2) = \begin{cases} 0 & \text{if } S(f(X_1), f(X_2)) \leq \tau, \\ 1 & \text{if } S(f(X_1), f(X_2)) \geq \tau. \end{cases}$$

This way, the problem of finding C amounts to find two functions $f : \mathcal{X} \to \mathbb{R}^N$ and $S : \mathbb{R}^N \times \mathbb{R}^N \to [0, 1]$ such that $S(f(X_1), f(X_2))$ is as close to 0 as possible whenever X_1 and X_2 have different forensic traces, and as close as possible to 1 whenever the two image patches share the same traces. Figure 7.2 shows the system overview.

In [162], Mayer and Stamm propose to design both the feature extractor function f and the similarity function S based on a learning strategy. In this section we specify the architecture as well as the training strategies developed in their work.

7.2.2 Architecture

The feature extractor is based on the MISLnet architecture [17] and is depicted in Figure 7.1. Namely, it consists of 5 convolutional blocks and 2 fully connected layers. Each of the convolutional layers, except for the first one, is followed by batch normalization, hyperbolic tangent activation and max-pooling. The size of the convolutional filters used at each layer are $5 \times 5 \times 3 \times 6$, $7 \times 7 \times 6 \times 96$, $5 \times 5 \times 96 \times 64$, $5 \times 5 \times 64 \times 64$ and $1 \times 1 \times 64 \times 128$ respectively. A stride equal to 1 is used in all the layers except for the second one, where the stride is set to 2. The last convolutional layer, which uses 1×1 kernels, can be regarded as a learned cross-feature maps associations. The max-pooling operation is performed using 3×3 kernels. The two fully connected layers that follow the convolutional blocks consist both of 200 neurons with hyperbolic tangent activation.

Two such feature extractor networks, in siamese configuration with weight-sharing, are used to process in parallel the two patches, producing a feature vector for each patch. Then, a similarity network takes both feature vectors as input and computes their similarity score. Figure 7.2 shows the complete system overview. The first layer of the similarity network

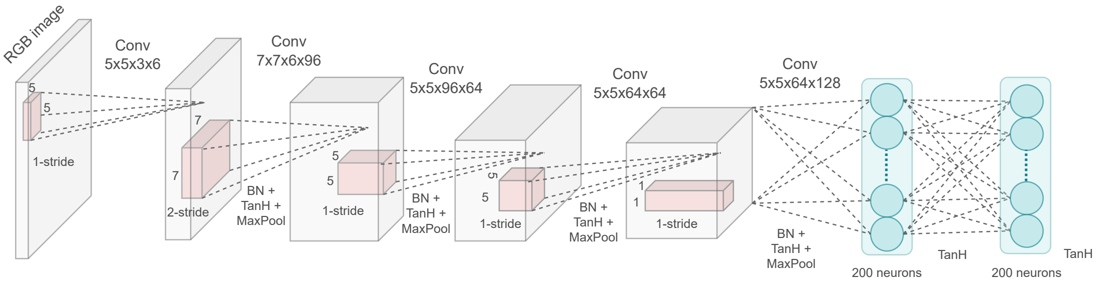


Figure 7.1: Feature extractor architecture.

consists of 2048 neurons with ReLu activation, that maps each feature vector into a new feature space. The authors use a hard-sharing siamese configuration for this first layer. Then, a new feature vector is constructed by concatenating both feature vectors and their element-wise multiplication. This vector feeds another fully-connected layer with 64 neurons. Finally, a single-neuron layer with sigmoid activation takes the resulting 64-dimensional vector and produces the similarity score associated to the pair of input patches.[1]

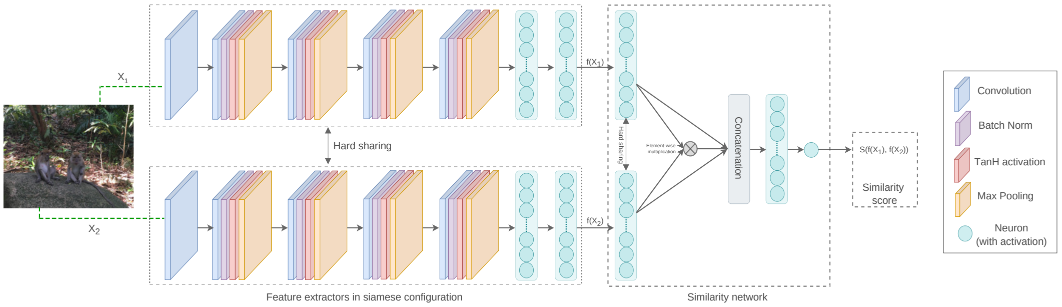


Figure 7.2: System overview. The first module consists of a pair of feature extractor networks in siamese configuration with weight-sharing. It takes two image patches and computes its corresponding feature vectors. These vectors are then compared by means of the second module (the similarity network), which computes a similarity score associated to the pair of image patches.

7.2.3 Dataset

Mayer and Stamm collected a dataset of 47,785 images from 95 different camera models. Among them, 26 camera models come from the Dresden dataset [81] while the rest are from the authors' database. This dataset is divided into three disjoint subsets. Subset 1 consists of 50 camera models selected randomly from those having at least 40,000 non overlapping 256×256 patches. Subset 2 comprises 30 camera models from the remaining ones having at least 25,000 non-overlapping patches. Subset 3 comprises the remaining 15 camera models. The complete list of camera models is given in Table 7.1. The camera models in blue were collected from the Dresden dataset [81].

Subsets 1 and 2 are used for training (see Section 7.2.4) while Subset 3 is used for evaluation, which we will not cover here. The interested reader is referred to the original paper [162].

[1] In practice, the authors use two output units with softmax activation, one indicating “similar" traces and the other one indicating “different" traces. The observed output for evaluation is the one corresponding to “similar".

Subset 1			
Apple iPhone 4	Agfa Sensor530s	Apple iPhone 4s	Canon EOS SL1
Apple iPhone 5	Canon PC1730	Apple iPhone 5s	Canon A580
Apple iPhone 6	Canon ELPH 160	Apple iPhone 6+	Canon S100
Apple iPhone 6s	Canon SX530 HS	Canon SX420 IS	Canon SX610 HS
Casio EX-Z150	Fujifilm S8600	Huawei Honor 5x	LG G2
LG G3	LG Nexus 5x	Motorola Maxx	Motorola Turbo
Motorola X	Motorola XT1060	Nikon S33	Nikon S7000
Nikon S710	Nikon D200	Nikon D3200	Nikon D7100
Panasonic DMC-FZ50	Panasonic FZ200	Pentax K-7	Pentax OptioA40
Praktica DCZ5.9	Ricoh GX100	Rollei RCP-7325XS	Samsung Note4
Samsung S2	Samsung S4	Samsung L74wide	Samsung NV15
Sony DSC-H300	Sony DSC-W800	Sony DSC-WX350	Sony DSC-H50
Sony DSC-T77	Sony NEX-5TL		
Subset 2			
Apple iPad Air 2	Blackberry Leap	Apple iPhone 5c	Canon Ixus70
Agfa DC-733s	Canon PC1234	Agfa DC-830i	Canon G10
Canon SX400 IS	Canon T4i	Fujifilm XP80	Fujifilm J50
HTC One M7	Kodak C813	Kodak M1063	LG Nexus 5
Motorola Nexus 6	Nikon D70	Nikon D7000	Nokia Lumia 920
Olympus TG-860	Panasonic TS30	Pentax OptioW60	Samsung Note3
Samsung Note5	Samsung S3	Samsung S5	Samsung S7
Sony A6000	Sony DSC-W170		
Subset 3			
Agfa DC-504	Canon Ixus55	Agfa Sensor505x	Canon A640
Canon Rebel T3i	LG Optimus L90	LG Realm	Nikon S3700
Nikon D3000	Olympus 1050SW	Samsung Lite	Samsung Nexus
Samsung Note2	Samsung S6	EdgeSony DSC-T70	

Table 7.1: Camera models used for training and evaluation. Camera models in blue come from the Dresden dataset [81]. Subset 1 is used during the first training phase. Subset 1 and 2 are used during the second training phase. Subset 3 is used for evaluation.

7.2.4 Training procedure

The system is trained in two phases. In the first phase, the feature extractor is trained by adding a fully connected layer with softmax activation. The feature extractor is trained as a source camera classifier, using image patches with associated labels corresponding to their source camera model. Research indicates that the deep features associated to camera model classification provide a good starting point for several forensics tasks [164]. The authors use a cross-entropy loss, optimized using stochastic gradient descent for 30 epochs, with batches of 50 images. Initially the learning rate is set to 0.001, and is halved every three epochs. The authors train two versions of the feature extractor: one using 128×128 patches and another one using 256×256 patches.

During the second training phase, the similarity network is trained to target a specific task. Here, the task is to determine if two image patches come from the same camera model, but it could be to determine a specific editing operation or a specific parameter given an editing operation. The labels in the training dataset are assigned accordingly: 0 indicates that the patches come from the same camera model, and a 1 indicates they come from different ones. During this phase, the weights of the feature extractor are fine-tuned, i.e. they are also updated to better fit the particular task. The similarity network is trained using stochastic gradient descent with cross-entropy loss for 30 epochs. The learning rate is initialized to 0.005, and then is halved every three epochs.

During the first training phase, the feature extractor is trained using 40,000 randomly sampled image patches from each of the camera models in Subset 1, giving a total of 2,000,000 image patches. During the second training phase, the similarity network is trained and the feature extractor weights are updated using a training dataset of pairs of image patches. This dataset is constructed using camera models in Subset 1 and Subset 2.

7.2.5 Demo

The goal of the method is to determine if a pair of images have been captured by the same camera model or not. It takes as input three images: a reference image, and two test images that will be compared to the reference one. To perform image-wise comparison built upon the patch comparison provided by the forensic similarity approach, the user is required to choose the number of randomly chosen patch-to-patch comparisons (ranging from 100 to 700) to be considered. The user can also decide if these patches are taken with overlap (half of the patch size) or not. Finally, the user can decide the patch size, being 128 and 256 the two available options.

The output of the demo is an interactive histogram showing the forensic scores obtained for each patch-to-patch comparison, for both image comparisons. By moving the mouse over the histogram, the user can recover the bins bounds as well as the count that corresponds to each bin. The user can also zoom in different sections of the histogram to better visualize the results.

7.3 Experiments

In this section we show several experiments conducted using the demo. The images used come from the Vision dataset [193] and the Forchheim dataset [93]. Notice that none of these datasets were used for training: all the images used for these experiments are new to the network.

To assess the performance of the forensic similarity approach, we designed three different experiments with different challenging scenarios. In the first experiment we test the approach using images coming from camera models that were used for training. In the next experiment we compare images coming from known models to images coming from unknown ones. Finally, we test the forensic similarity approach on pairs of camera models that are unknown to the network.

All the experiments were conducted using the default parameters values. Namely, the number of patch-to-patch comparisons is 300, the size of the patches is set to 256 and the patches are taken without overlap.

7.3.1 Known camera models

Figure 7.3 shows the results obtained when applying the forensic similarity approach for source camera comparison for images taken with camera models that are known to the network. Namely, for this experiment we use camera models that are included in Subset 1 and Subset 2 (see Table 7.1).

We observe that, when faced to known camera models, the similarity scores between images coming from the same camera model concentrate around 1. Furthermore, the similarity scores obtained when comparing images coming from different camera models concentrate around 0, except for the case in which iPhone 6 is compared to iPhone 6s. This might be mainly due to the fact that these two devices share similar processing pipelines. In this case we observe that the similarity scores wrongly concentrate around 1. However, it can be also noticed that the non-matching histogram exhibits a thicker tail than the matching one.

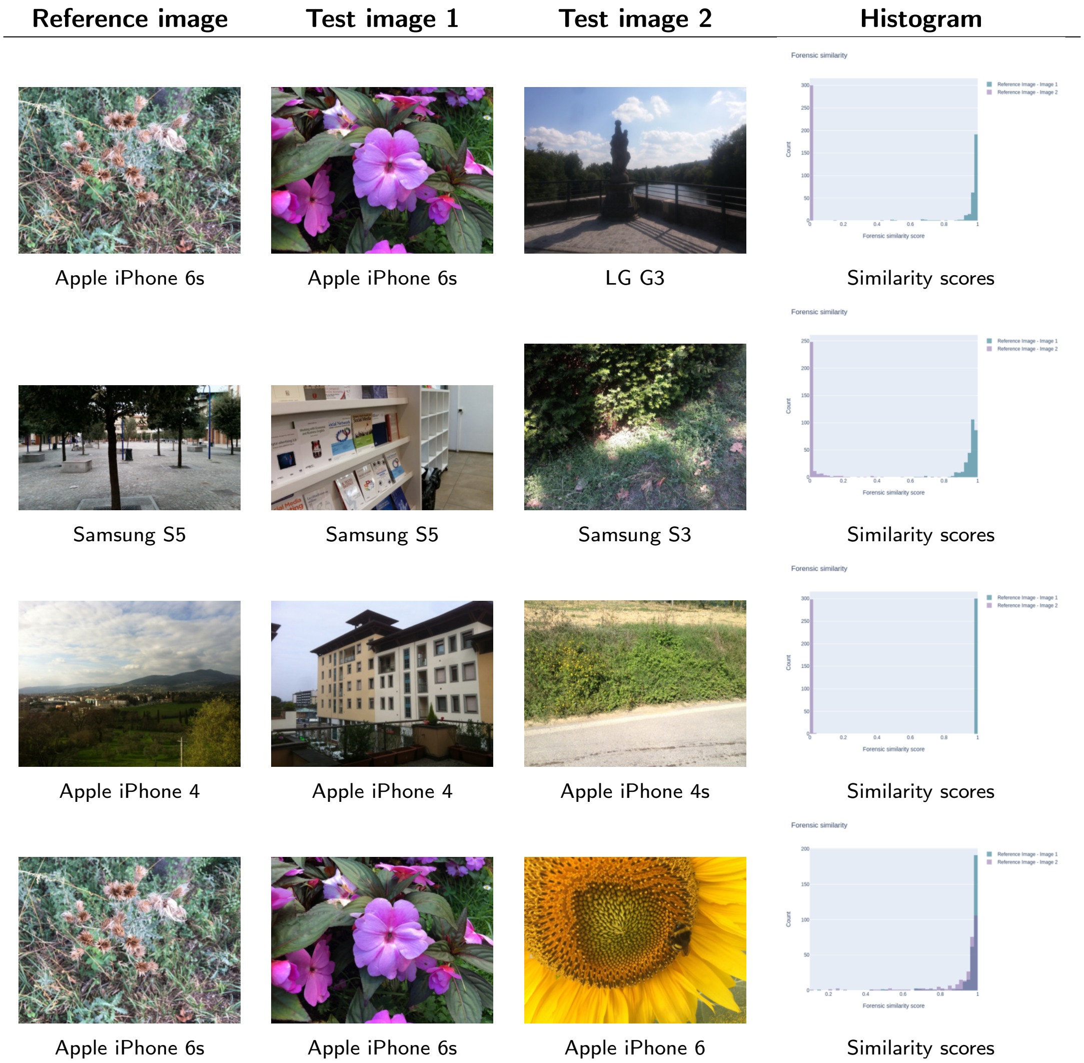


Figure 7.3: Results of the forensic similarity approach applied to source camera comparison when images under test come from camera models used during training.

7.3.2 Known and unknown camera models

Figure 7.5 shows the results obtained when applying the forensic similarity approach for source camera comparison, to test images taken with camera models that are known to the network against images from camera models that were not used for training. Namely, for this experiment we test camera models that are included in Subset 1 and Subset 2 (see Table 7.1) against camera models that are not part of them.

Under this setting the results are more heterogeneous. The network is able to distinguish images coming from iPhone 6s from those coming from Huawei P9 Lite as well as those coming from Samsung S3 Mini and Samsung S3. When comparing iPad Mini to iPhone 4, the network also delivers similarity scores close to 0. However, the network is not able to identify two images taken with iPad Mini as having similar forensic traces. On the other hand, the results obtained when comparing Samsung S5 and Sony Xperia E5, the histogram shows that the patch-to-patch similarity scores seem uniformly distributed over the $[0, 1]$ interval, therefore preventing from taking any conclusion about their forensic similarity.

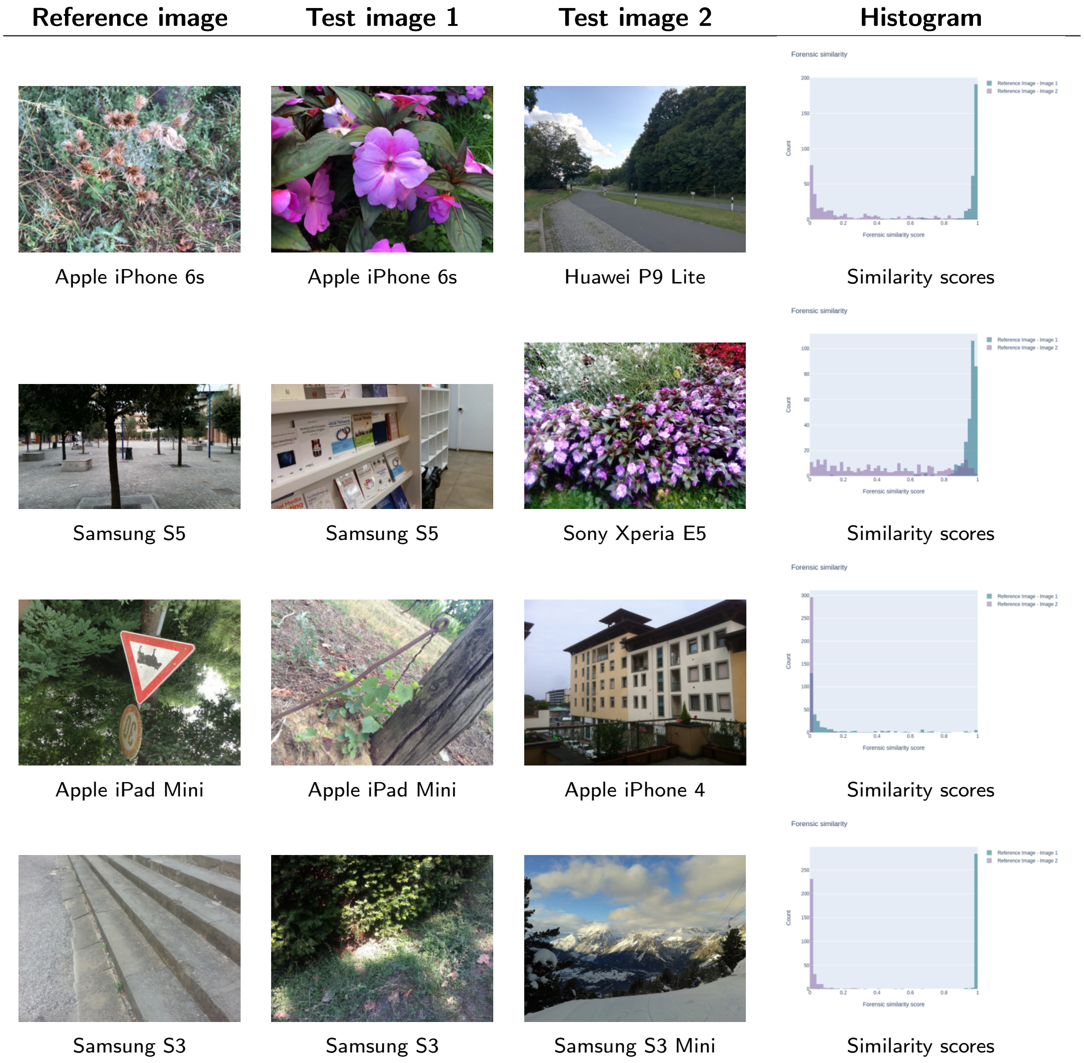


Figure 7.4: Results of the forensic similarity approach applied to source camera comparison, when comparing images coming from camera models used during training to images from camera models unknown to the network.

7.3.3 Unknown camera models

Figure 7.5 shows the results obtained when applying the forensic similarity approach for source camera comparison to test images taken with camera models that are unknown to the network.

Namely, in this experiment we consider camera models that are not included in Subset 1 or Subset 2 (see Table 7.1).

In this challenging scenario, we observe degraded results with respect to the previous experiments. Indeed, camera models showing good results for matching images (Motorola Z2 Play and Huawei P9 Lite), the mismatching results are incorrect. On the contrary, the devices delivering good results when compared to a different camera model (Google Pixel 3 and Sony Xperia E5), fail at identifying matching images.

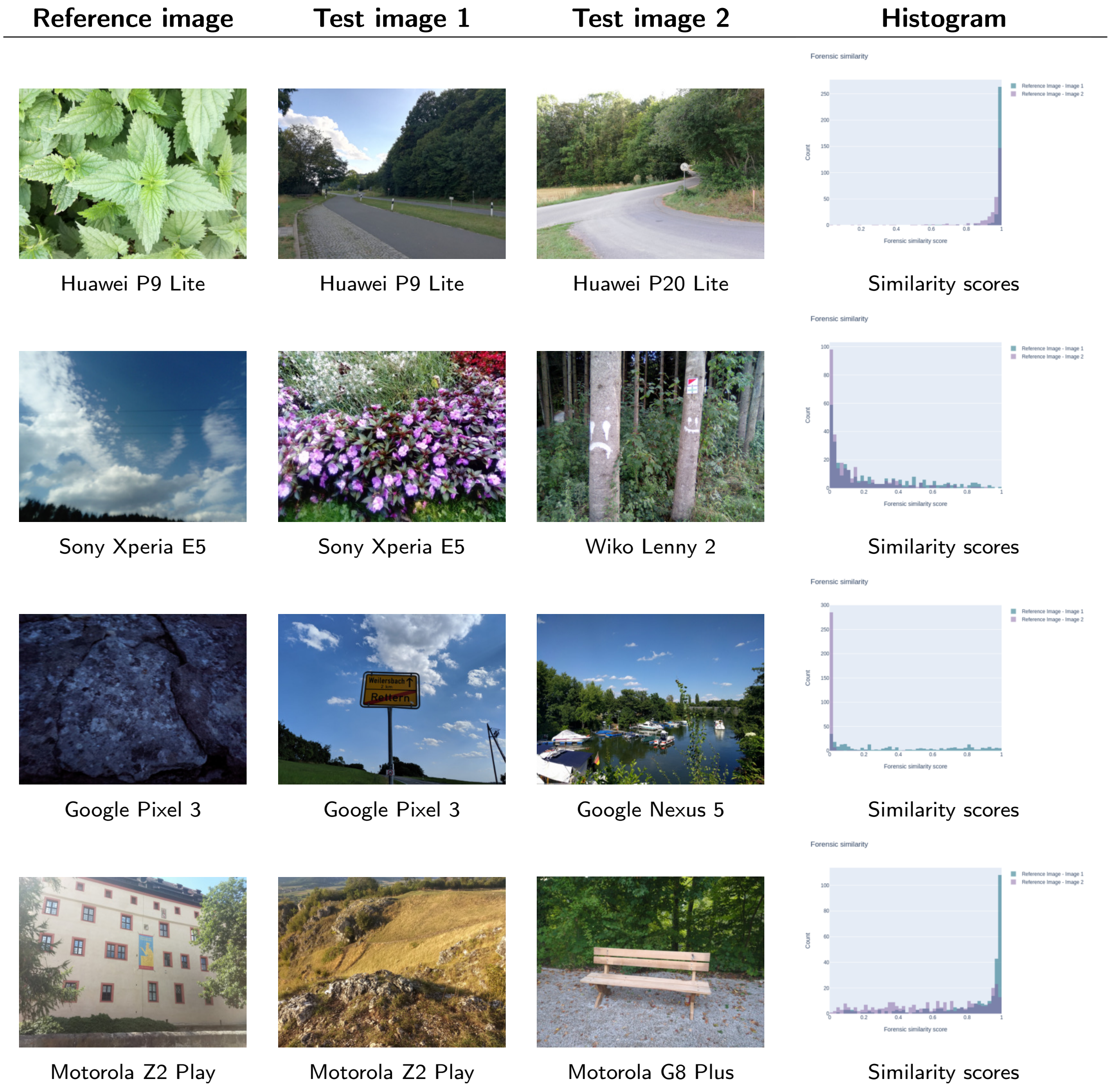


Figure 7.5: Results of the forensic similarity approach applied to source camera comparison, when comparing images coming from camera models that were not used for training.

7.4 Conclusion

In this Chapter we explored the use of the forensic similarity approach (Chapter 5) for source camera model comparison. Such an approach aims at determining whether two image patches share the same forensic traces or not. In addition, we provide an online demo at: hhttps://ipolcore.ipol.im/demo/clientApp/demo.html?id=424. The method seems to be effective when, at least on of the images, comes from a camera model used during training. Still, there are some exceptions to this, such as when comparing images coming from Apple iPhone 6 and Apple iPhone 6s. This example points to critical point: even if these are different mobile models, is this enough to guarantee that the cameras are not the same? Besides this, the method seems not to generalize correctly to unseen camera models.

Esta página ha sido intencionalmente dejada en blanco.

Chapter 8

Photo-response non-uniformity

The problem of detecting the presence of a PRNU pattern in a query image can be stated as a hypothesis testing problem. The test statistics that have been proposed to perform this test all suffer from the same drawback: decision thresholds need to be set empirically. This poses a major problem for source camera certification, since these methods do not provide an accurate false alarm rate for each detection but rather a lower bound related to the size of the dataset used to derive such thresholds. In this chapter, we propose an alternative approach for source camera certification that can be used together with the classic testing strategies. Our method relies on two hypothesis tests based on local correlations which do not require computing empirical distributions. The p-value of the test serves is a statistically founded confidence measure that can serve as certification. Our results show that in most cases, the PRNU true detections give almost absolute guarantees, with p-values smaller than 10^{-100}, while most true negatives deliver p-values above 10^{-1}.

The work presented in this chapter is published as *PRNU-based source camera statistical certification* in the IEEE International Workshop on Information Forensics and Security (WIFS) 2023 [75].

8.1 Introduction

Digital cameras are equipped with sensors that count the number of incident photons and output the corresponding voltage. The two main technologies used in camera sensors, CCD and CMOS, are both two-dimensional arrays of photosensitive cells, each corresponding to a pixel. Due to physical imperfections, digital sensors leave unique traces in the image which can be used for several forensics tasks.

One of these traces is the *photo response non-uniformity* (PRNU) pattern. It is caused by a non-uniform response of each pixel to the same amount of incoming photons, due to manufacturing imperfections. In contrast to other sources of noise, the PRNU pattern is deterministic, and it is systematically present in every image captured with a given sensor.

Since each sensor array produces a unique PRNU pattern, it can be considered as a device fingerprint [151]. This observation has led to several forensic applications: source camera identification [31, 151], image and video forgery detection [31, 121, 170], scanner identification [82, 115] - even CT scanners [116] -, deepfakes and computer generated graphics detection [150, 167], and profile linking [18].

In this work we are interested in digital source camera authentication. Given an image and a camera, we want to certify if the image was taken with the camera in question. To address this problem, we assume that the camera, or at least a certain amount of images acquired with it, are available.

The standard procedure to estimate the PRNU of a given camera is to average the noise residuals of a certain amount of images captured by said camera [31, 129, 151]. The noise residuals are generally extracted using a denoising filter [31, 46, 136, 151]. Since these noise residuals contain other types of noise and random variations in addition to the fixed PRNU, these residuals are averaged to suppress the random variations and enhance the fixed pattern that is present in all of them. This fixed pattern can be then refined to discard non-unique artifacts that are not part of the PRNU.

Once the PRNU pattern is estimated, the problem of source camera identification consists in determining if the PRNU pattern is present in the query image or not. This problem can be stated as a statistical hypothesis testing problem, with the null hypothesis corresponding to the absence of a tested PRNU pattern, and the alternative its presence [31, 83]. The test statistic used to decide between one alternative or the other is usually a correlation metric [83, 151]. This kind of metric provides a measure of the presence of the PRNU of the camera in the given image. By comparing the observed test statistic to a pre-fixed threshold that depends on the significance level at which the test is performed, one can conclude if the query image was taken with the camera under investigation or not.

PRNU analysis for source camera identification delivers robust results that are, furthermore, stable over time [84, 151]. Its reliability has led to its acceptance as court evidence. Indeed, PRNU-based source camera identification has passed the Daubert standard [123]. In order to be admissible in court, a scientific technique is required—among other conditions—to provide an error rate. This condition aims at enforcing methods delivering statistically validated results.

A major drawback, shared by all the test statistics proposed for PRNU detection, is that their distribution needs to be determined empirically by analyzing the behaviour of a given camera's PRNU pattern with respect to images acquired using the same device and to images acquired with a different device. This poses a major problem for source camera certification since these methods do not provide an accurate false alarm rate for each detection, but rather a lower bound related to the size of the dataset used to derive such thresholds.

In this section, we propose an alternative strategy for source camera certification that can be used in conjunction with the classic testing approaches. Our method relies on two hypothesis tests based on local correlations that do not require computing empirical distributions. For each detection, we provide the p-value of the test as a confidence measure. As shown in Figure 8.1, in most cases PRNU true detections are almost absolute, with p-values smaller than 10^{-100}. On the other hand, most true negatives deliver p-values above 10^{-1}.

8.2 Related work

8.2.1 PRNU estimation

There are two key choices in the PRNU estimation procedure: the choice of the denoising filter used to extract the noise residuals, and the residuals merging procedure.

Early methods [31, 151] are built on a wavelet-based denoising filter. Namely, both works by Chen et al. and Lukas et al. use the Mihcaks filter [166] to extract the noise residual of each image. With the development of new and more sophisticated denoising algorithms, subsequent works, such as [136] and [46], proposed to modify the original algorithm by choosing another denoising filter. In this sense, the performance of methods such as BM3D [51] and the Argenti

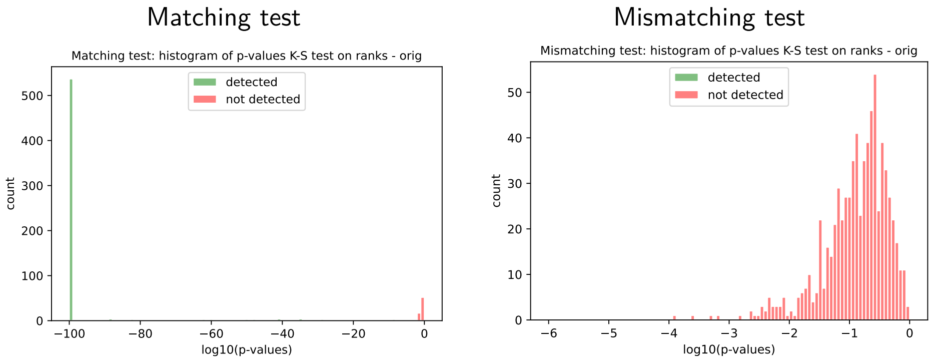


Figure 8.1: Histograms of the $\log_{10}(\text{p}-values)$ obtained with the Kolmogorov-Smirnov test on the uniformity of the ranks. The histogram in the left corresponds to the matching test and the one on the right to the mismatching test in the native resolution images from Forhheim dataset [93]. The matching histograms are truncated in -100 and, therefore, all the p-values below this bound contribute to this bin. We observe that the proposed approaches deliver very significant detections. Furthermore, the p-values obtained for the mismatching test are mostly above 10^{-1}.

filter [7] were tested. These results suggest that BM3D slightly outperforms the previous formulation of the algorithm but with a much higher computational cost.

Other denoising techniques such as context adaptive interpolation [111, 217], adaptive spatial filtering [44], content adaptive guided image filtering [231] and the total variation filter [79] were also addressed in the literature. Furthermore, the use of CNN denoisers such as DnCNN [235], FFDNet [236], DANet [228] and ADNet [208] for PRNU extraction has also been proposed [233]. In this case, denoisers need to be specifically trained for the task.

The simplest merging method consists in averaging the noise residuals [151]. Chen et al. [31] proposed to estimate the PRNU using a maximum likelihood estimator (MLE), which takes into account the fact that PRNU is a multiplicative factor. Lawgaly et al. [129] point out that the noise variance changes from one image to another depending on the lighting conditions and camera settings. They therefore propose to perform a weighted average of the noise residuals.

Regardless of the choice of the denoiser and the averaging procedure, the estimated pattern does not only contain the PRNU pattern but also other non-unique artifacts that are systematically present in every image. These artifacts cannot be used for source camera identification since they might be shared between several devices. To further refine the estimation, several enhancement techniques have been proposed. Chen et al. [31] suggested removing them by performing two operations: removing the linear pattern and using a Wiener filter in the Fourier domain. Other enhancing techniques such as the phase-only operation [110] and spectrum equalization [141] have also been proposed. However, according to a systematic evaluation [2], the only post-processing technique that yields significant improvements is the removal of shared components suggested by [31].

8.2.2 PRNU detection

Once the PRNU pattern P is extracted, the problem of detecting its presence in a given image is formulated as a hypothesis test where the null hypothesis is its absence and the alternative its presence. A test statistic is needed in order to perform the statistical test.

The pioneer work of Lukas et al. [151] used the Pearson correlation between the estimated pattern P and the noise residual R extracted from the query image as a test statistic:

$$\rho(P,R) = \frac{(P-\overline{P})\cdot(R-\overline{R})}{||P-\overline{P}||||R-\overline{R}||}, \tag{8.1}$$

where the over-line denotes the mean value. After attempting to model theoretically the distribution of this test statistic, they conclude that the only feasible solution is to set the decision threshold empirically.

Chen et al. [31] pointed out that textures and denoising imperfections may affect the detection of the PRNU in the noise residuals. The authors constructed a correlation predictor that takes into consideration these observations. Some modern approaches to this construction have also been addressed in the literature [27]. Liu et al. [145] followed the same approach but instead of using the whole image to compute the correlation, they choose to use only the blocks having the highest signal-to-noise ratio, where the signal here refers to the PRNU.

The Pearson correlation coefficient presents an important limitation: with the presence of non-unique artifacts, its value raises and produces false positives. To address this issue, Goljan [83] proposed to use the peak-to-correlation energy (PCE), which suppresses the effect of periodic patterns,

$$\mathsf{PCE}\,(P,R) = \frac{\rho^2(s_{\mathsf{peak}};P,R)}{\frac{1}{mn-|\mathcal{N}|}\sum\limits_{s\notin\mathcal{N}}\rho^2(s;P,R)}, \tag{8.2}$$

where the correlation coefficient $\rho(s;P,R)$ is computed at the spatial shift $s=(s_1,s_2)$, $s_{\mathsf{peak}} = \arg\max \rho(s;P,R)$, $\mathcal{N}$ is a square neighborhood around s_{peak} and (m,n) is the size of image.

Still, with this measure the information about the sign of the correlation is lost. To preserve it, a signed version of the PCE can be used instead:

$$\mathsf{sPCE}\,(P,R) = \frac{\rho^2(s_{\mathsf{peak}};P,R)\times \mathsf{sign}(\rho(s_{\mathsf{peak}};P,R))}{\frac{1}{mn-|\mathcal{N}|}\sum\limits_{s\notin\mathcal{N}}\rho^2(s;P,R)}, \tag{8.3}$$

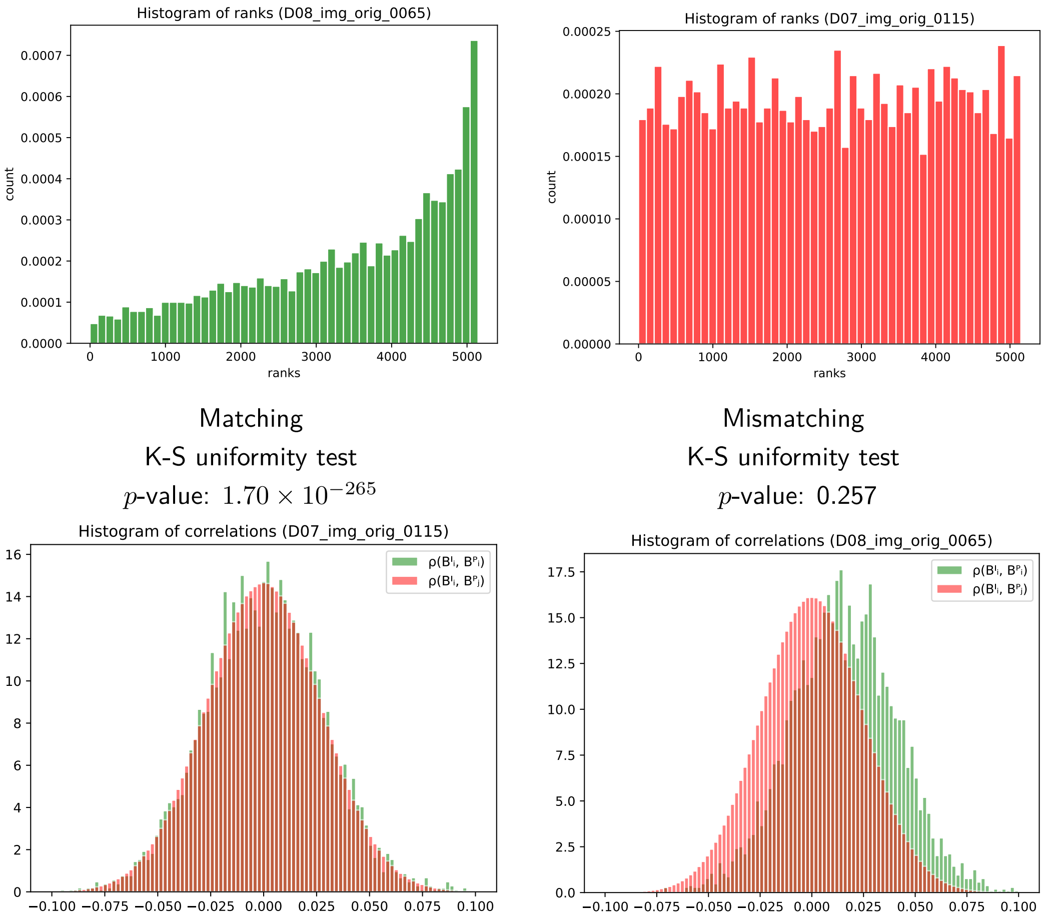
or, equivalently, the correlation over circular cross-correlation norm (CCN) introduced by Kang et al. [110],

$$\mathsf{CCN}\,(P,R) = \frac{\rho(s_{\mathsf{peak}};P,R)}{\sqrt{\frac{1}{mn-|\mathcal{N}|}\sum\limits_{s\notin\mathcal{N}}\rho^2(s;P,R)}}. \tag{8.4}$$

These metrics are more robust to the presence of non-unique periodic patterns. However, decisions thresholds remain empirical. Indeed, despite the theoretical development done in [84], the authors found that the data did not follow the theoretical distribution. Therefore, they ended up computing decision thresholds empirically. They found that a threshold of 60 guaranteed a false alarm rate of 10^{-6}. However, this threshold depends on the JPEG-quality of the images [85].

8.3 New source camera statistical certification

PRNU analysis has shown an exceptional performance for source camera identification and has even been admitted as scientific evidence in court rooms. However, as reviewed in Section 8.2.2, PRNU detection metrics are unable to deliver accurate false alarm rates for each detection.



Matching
K-S uniformity test
p-value: 1.70×10^{-265}

Mismatching
K-S uniformity test
p-value: 0.257

Matching
K-S two samples test
p-value: 2.24×10^{-262}

Mismatching
K-S two samples test
p-value: 0.307

Figure 8.2: **Top**: histograms of ranks of an image containing the PRNU pattern (left) and of an image which does not (right). In the matching case, the density is higher for larger rank values. On the contrary, for the mismatching case, the ranks follow an uniform distribution. **Bottom**: normalized histograms of correlations. In green $\{\rho(B_i^I, B_i^P) : i = 1, \ldots, K\}$ and $\{\rho(B_i^I, B_j^P) : j \neq i\}$ in red. The left plot corresponds to an image containing the PRNU pattern and the right one to an image that does not contain it. For the matching case, the mode of the green histogram is bigger than the one of the red histogram. On the other hand, in the non-matching case both histograms coincide.

Our proposed testing approach does not require the construction of such a dataset to model the test statistics under the null hypothesis. Instead, we learn a stochastic background model from the image itself. Our approach can help certify most of the results given by the classic metrics, but without the need for empirical thresholds.

Let I be a probe image and P the PRNU pattern extracted from the candidate source camera C. We want to determine whether the image I was taken with camera C or not. Our null hypothesis is that image I was not taken with camera C and hence, the PRNU pattern P is not present:

$$\begin{aligned} &H_0)\ P \text{ is not present in } I, \\ &H_1)\ P \text{ is present in } I. \end{aligned} \tag{8.5}$$

for testing, we consider $b_1^I, \ldots, b_K^I$ blocks without overlap of size $W \times W$ extracted from the noise residual of image I and $b_1^P, \dots, b_K^P$, $W \times W$ blocks without overlap extracted from the PRNU pattern, where W is a fixed hyperparameter. For a given correlation metric ρ and for each block b_i^I, we compute its correlation $\rho\left(b_i^I, b_j^P\right)$ with b_j^P for $j = 1, \dots, K$.

If C is the source camera of I, for each b_i^I we expect to observe that $\rho(b_i^I, b_i^P)$ exhibits higher values than $\rho(b_i^I, b_j^P)$ for $j \neq i$. Conversely, if C is not the source camera, the behaviour of $\rho(b_i^I, b_i^P)$ should be similar to that of $\rho(b_i^I, b_j^P)$ for $j \neq i$.

We propose two testing approaches to capture this effect. The first is based on the rankings of the correlations $\rho(b_i^I, b_i^P)$ with respect to the rest of the correlations $\rho(b_i^I, b_j^P)$, $j \neq i$. The second consists in directly comparing the distribution of the correlations $\rho(b_i^I, b_i^P)$ to the rest.

Here we have focused on the case where there is no shift between the PRNU pattern and the image. However, the proposed approach can be adapted to shifted images by searching for the offset parameter for which the correlations deviate from the background model.

8.3.1 Tests based on ranks

For each block $b_1^I, \dots, b_K^I$ in the image's residual consider the variables $r_1^I, \dots, r_K^I$ defined as the rank of $\rho(b_i^I, b_i^P)$ amongst all the correlations $\rho(b_i^I, b_j^P)$ for $j = 1, \dots, K$, in ascending order. Under the null hypothesis, $r_1^I, \dots, r_K^I$ follow a discrete uniform distribution with values $1, \dots, K$. However, if the PRNU pattern is present, $r_1^I, \dots, r_K^I$, will concentrate more probability in the higher values. This situation is depicted in Figure 8.2 (top). Following this observation, we can reframe the hypothesis test 8.5 as:

$$\begin{aligned} &H_0) F_{r_1^I, \dots, r_K^I} = F_{\mathcal{U}[\{1,\dots,K\}]}, \\ &H_1) F_{r_1^I, \dots, r_K^I} < F_{\mathcal{U}[\{1,\dots,K\}]}, \end{aligned} \tag{8.6}$$

where F is the cumulative distribution function and $\mathcal{U}\left[\{1, \dots, K\}\right]$ corresponds to a discrete uniform distribution with values $1, \dots, K$. Note that the alternative hypothesis captures only the case where the deviation from uniformity comes from a probability concentration on high rank values. The one-tailed nature of this test delivers lower p-values than with two-tailed tests.

To perform this hypothesis test there are several alternatives. Here we use one of them: the classic one-sided Kolmogorov-Smirnov test. This test compares the two distributions in Equation 8.6 with a statistics based on the supremum distance.

Figure 8.2 shows the p-values obtained with the proposed test. It accurately detects the deviations from uniformity and, therefore, detects the presence of the PRNU pattern. On the other hand, the test is not able to reject H_0, which is consistent with the fact that these ranks come from an image that does not contain the tested PRNU pattern.

8.3.2 Tests on the correlation distributions

The second approach we developed consists in establishing a background model for the non-matching correlations. To do so, we consider the correlations $\{\rho(b_i^I, b_j^P) : j \neq i\}$. This framework shares the same idea as previous approaches where the distributions of the correlations under H_0 were computed empirically but, instead of using images from other devices, we construct this background model from the image itself.

Once the background distribution is established, we compare it to the distribution of $\{\rho(b_i^I, b_i^P) : i = 1, \dots, K\}$. We expect both distributions to be the same for the non-matching case whereas, for the matching case, $\{\rho(b_i^I, b_i^P) : i = 1, \dots, K\}$ should present higher values. This situation is depicted in Figure 8.2 (bottom). For the matching case, we observe that

		sPCE	K-S uniformity test	K-S two-samples test
Forchheim Dataset	Native	**646 74**	645 75	**646 74**
	Twitter	579 141	**596 124**	590 130
	Telegram	263 457	**272 448**	262 458
	WhatsApp	**260 460**	248 472	243 477
	Facebook	**32 688**	26 694	25 695
	Instagram	**21 699**	17 703	16 704
VISION Dataset	Native	7007 405	**7059 353**	7057 355
	Facebook Highh	**6948 617**	6911 654	6885 680
	Facebook Low	**4456 3109**	4384 3181	4280 3285
	WhatsApp	7139 426	**7194 371**	7182 383

Table 8.1: True positives (TP) and false negatives (FN) for each method on each social media dataset when setting a threshold of 60 on the sPCE and its equivalent p-value threshold of 10^{-6} on the p-values of the proposed testing approaches.

the distributions are visibly different. In contrast, for the non-matching case, the distributions behave similarly.

To test whether $\{\rho(b_i^I, b_j^P) : j \neq i\}$ and $\{\rho(b_i^I, b_i^P) : i = 1, \ldots, K\}$ behave similarly we perform a Kolmogorov-Smirnov test with two samples. The hypothesis are:

$$\begin{aligned} &H_0) F_{\{\rho(b_i^I, b_j^P) : j \neq i\}} = F_{\{\rho(b_i^I, b_i^P) : i=1,\ldots,K\}}, \\ &H_1) F_{\{\rho(b_i^I, b_j^P) : j \neq i\}} > F_{\{\rho(b_i^I, b_i^P) : i=1,\ldots,K\}}. \end{aligned} \tag{8.7}$$

This test measures the distance between the two cumulative distribution functions using the supremum distance. Instead of using a theoretical distribution as we did before, we are comparing two empirical distributions.

In Figure 8.2 (bottom) we present the p-values obtained with this Kolmogorov-Smirnov test. Note that this test is able to accurately detect the deviations from one distribution to the other in the case shown in the left. However, the test on the right does not reject H_0.

8.4 Experimental analysis

8.4.1 Datasets.

To test the performance of the proposed PRNU detection testing approaches we used the Forchheim Dataset [93] and the VISION dataset [193]. Both datasets include different qualities: not only the native resolution images is available but also their social media versions.

In the case of the Forchheim Dataset [93], the available social media versions are Facebook, Twitter, Telegram, WhatsApp, and Instagram. In the case of the VISION Dataset [193], the available social media versions are WhatsApp, Facebook (low) and Facebook (high).

	Native			WhatsApp			Instagram			Telegram			Twitter			Facebook		
W	48	96	128	48	96	128	48	96	128	48	96	128	48	96	128	48	96	128
K-S test on ranks	645	645	**647**	**275**	248	253	16	**17**	14	**285**	272	264	595	**596**	593	**30**	26	25
K-S test on corrs	645	**646**	645	**274**	243	243	14	**16**	12	**279**	262	255	**593**	590	592	**30**	25	24

Table 8.2: True positives (TP) for each testing approach on each social media from the Forchheim dataset [93] when varying the size of the blocks $W \times W$ used to compute the local correlations.

8.4.2 PRNU estimation.

To estimate the PRNU patterns we followed the well-known method proposed by Chen et al. [31]. In their work, they use a maximum likelihood estimator (MLE) to extract the PRNU residuals from a certain amount of noise residuals of images captured by the said camera.To extract the noise residuals the authors use the Mihçak's filter [166].

For the Forchheim dataset, we estimated the PRNU pattern using 50 reference images selected as having the same image orientation (portrait). For social networks, we used the version of the reference images that corresponded to the media.

The VISION dataset has flat images of each device. These images were used to extract the PRNU. However, they are only available at the native resolution. For the social media versions, we proceeded as for the Forchheim dataset, by selecting 50 reference images to estimate the PRNU pattern.

8.4.3 Matching and Mismatching tests.

With the estimated PRNU patterns, we conducted matching and mismatching tests. The matching test consisted in comparing the estimated PRNU patterns to the single-image PRNU patterns extracted from images coming from the same device and same social network. For the Forchheim dataset, the matching test was conducted on 30 images randomly chosen amongst those that were not used for PRNU estimation. For the VISION dataset, we used for each device all the images not used to extract the reference pattern. The number varies for each device.

The mismatching test consisted in comparing the estimated PRNU patterns to the single-image PRNU patterns extracted from images coming from different devices and same social network. For the Forchheim dataset, we designed a more challenging mismatching test by choosing 30 test images amongst devices of the same brand. For the VISION dataset, we randomly selected the same number of images used in the matching test but from other devices than the one under analysis. In both cases these images were selected amongst those that were not used to extract the PRNU pattern.

8.4.4 Performance assessment.

To assess the performance of the testing approaches presented in Section 8.3, we used the sPCE metric (Equation 8.3), which is the preferred test statistic in source camera identification applications [2].

For each query image, we tested the possible rotations that aligned the image orientations and kept the highest sPCE score and the smallest p-value for each test. Since there is no shift

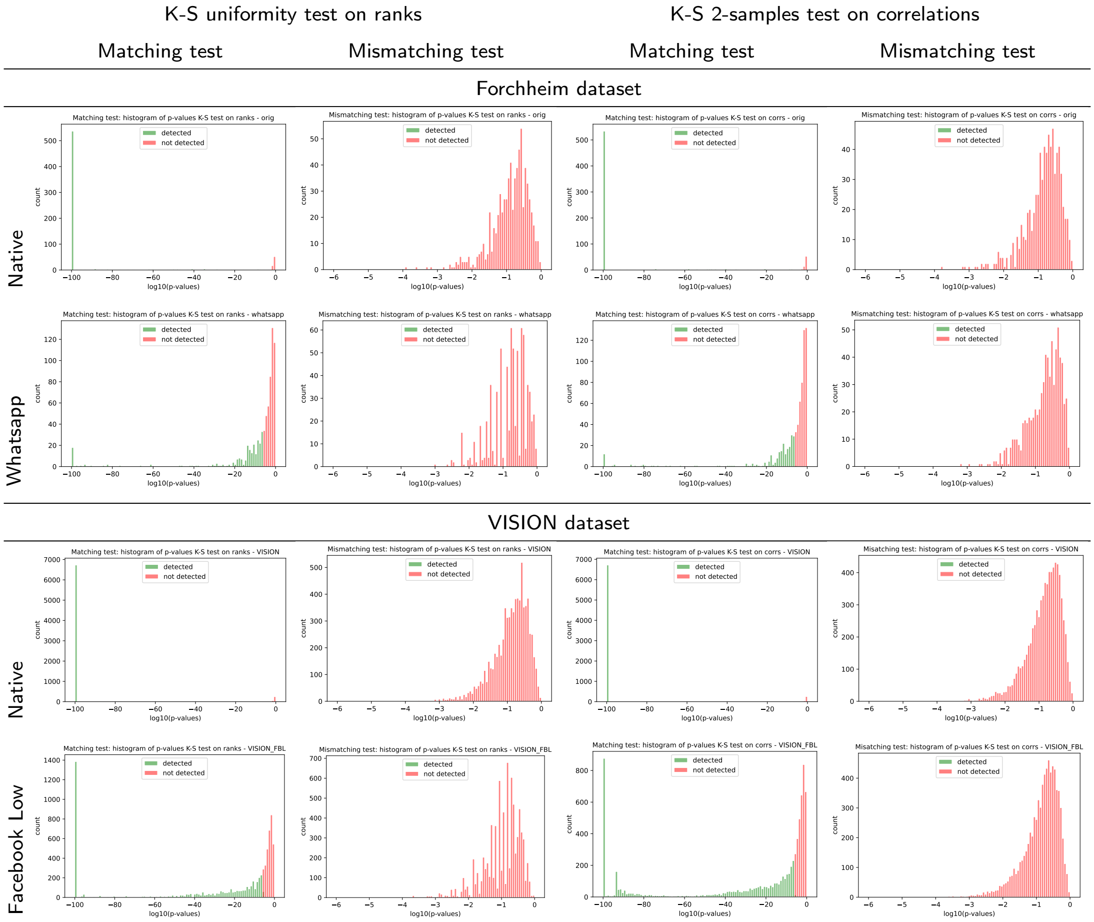


Figure 8.3: Histograms of the $\log_{10}(\text{p}-values)$ obtained in the matching and mismatching tests in the Forchheim dataset (Native and WhatsApp) and in the VISION (Native and Facebook Low) dataset [193], for each testing approach. The matching histograms are truncated in -100 and, therefore, all the p-values below this bound contribute to this bin. We observe that the proposed approaches deliver very significant detections for all the image versions. Furthermore, the p-values obtained for the mismatching test are far from the 10^{-6} threshold, most of them are even above 10^{-1}.

between images, in the sPCE computation s_{peak} was directly set to $(0, 0)$. We used blocks of size 96×96 and, as correlation metric, the correlation coefficient defined in Equation 8.1.

To assess the performance of the proposed methods, we established a threshold of 60 for the sPCE as done in [84]. According to the authors, they tested 1,024,050 mismatching images and got a maximum sPCE value of 57. Therefore, they conclude that a threshold of 60 guarantees a false alarm rate smaller than 10^{-6}. Hence, we could set a p-value threshold of 10^{-6} in our methods for a fair comparison.

When setting such thresholds, none of the methods deliver false detections in any of the datasets. This means that none of the images in the mismatching test is wrongly detected as having each tested PRNU pattern. This result is consistent with the threshold set on the false alarm rate.

Regarding the matching test, the results when setting these thresholds are shown in Table 8.1. Firstly, we observe that the percentage of detections is smaller in all the social media versions of the images, than the one obtained with its native resolution. Also, we observe that both Kolmogorov-Smirnov tests reach a similar performance, with slightly better results with the Kolmogorov-Smirnov uniformity test on ranks.

The proposed approach and the sPCE statistic perform similarly. In some cases, such as the WhatsApp, Instagram and Facebook images from the Forchheim dataset and both of the Facebook versions of the VISION dataset, the sPCE metric delivers more detections than our approaches. This does not mean that our tests do not work but rather that the level of confidence of those detections are above the threshold of 10^{-6}.

On the other hand, our approach gains some detections that are missed with the classic test statistics. Such is the case of Twitter and Telegram images from the Forchheim dataset and the native and WhatsApp images from the VISION dataset.

The key point of our testing approach is that we can obtain the false alarm probability—by means of the p-value—for every detection. Indeed, the magnitude of the sPCE, whenever bigger than 60, only tells us that the false alarm rate associated with the detection is smaller than 10^{-6}. However, we are left in the ignorance of the exact false alarm rate corresponding to the observed value. By computing the p-value, we directly obtain the false alarm probability of each detection.

Figure 8.3 shows the histograms of the $\log_{10}(p\text{-values})$ obtained in the Forchheim dataset (Native and WhatsApp) and in the VISION (Native and Facebook Low) dataset [193], for each testing approach.

Besides the possibility of setting a threshold and obtaining a binary classification, we can also compute the false alarm rate associated to each detection which, in many cases, exceeds the settled threshold, making the detection more meaningful. The number of detections (see Table 8.1), as well as their meaningfulness, decrease when considering social medias that degrade image quality.

Regarding the mismatching test, the p-values obtained are far from the 10^{-6} threshold, most of them are even above 10^{-1}. Furthermore, this observation is not nuanced when considering the social network versions of the images. Indeed, even for these cases mismatching images deliver high p-values, far from the settled threshold.

8.4.5 Empirical check of the probability of false alarm.

So far we have presented the theoretical guarantees given by the proposed approach. Still, it is important to check if such theoretical probabilities of false alarm actually match the empirical ones. Figure 8.4 presents the empirical false alarm rates, plotted against the theoretical ones. We observe that, though empirical false alarm rates are bigger than the theoretical ones, this difference seems to be constant regardless the magnitude of such probabilities. We conclude that, despite that the actual theoretical value does not match the empirical probabilities, they do match in terms of order of magnitude.

8.4.6 Influence of the block size.

The proposed testing approaches have one hyperparameter, W, which sets the size of the blocks ($W \times W$) used to compute the local correlations. The bigger these blocks are, the more reliable the correlation estimation is. Conversely, the smaller it is, the more samples needed to perform the Kolmogorov-Smirnov test.

The results on the Forchheim dataset when varying the block size are presented in Table 8.2. While PRNU detection on native resolution images works better using bigger block sizes, the performance on social networks is better when using smaller blocks. Also, when using blocks of size 48×48 on WhatsApp images the proposed approach outperforms sPCE (Table 8.1).

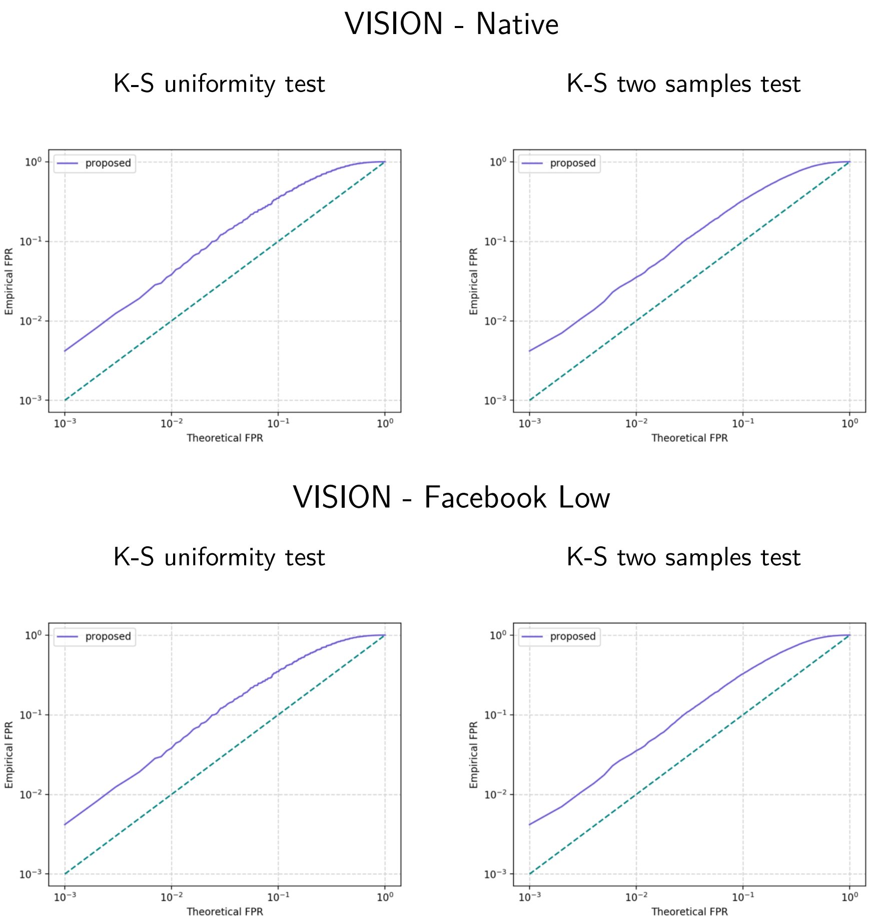


Figure 8.4: Empirical false alarm rates plotted against the theoretical ones for VISION dataset in both, Native and Facebook Low resolutions. We observe that, though empirical false alarm rates are bigger than the theoretical ones, this difference seems to be constant regardless the magnitude of such probabilities.

These results suggest that a multi-testing approach where several block sizes are considered or an adaptive block size that depends on the image size could improve the proposed approaches. This path will be pursued in future works.

8.5 Conclusion

We presented a new testing framework for PRNU detection in source camera certification tasks. These tests are constructed on local correlations between the query image noise residual and the PRNU extracted from the suspected camera. We presented a way of modeling the ranks of these correlations and a background model for the correlations themselves that can be learnt from the image itself. We showed that our approach complements the classic testing statistics by associating a very informative confidence measure with detections and rejections. In some cases a PRNU presence was detected in images that were missed by the previous existing methods.

Esta página ha sido intencionalmente dejada en blanco.

Part III

Counter-forensics

Chapter 9

A study of CamTE: a Camera Trace Erasing Network

As pointed out in Part I and Part II, camera traces - though imperceptible to the naked eye - play a crucial role in various forensic tasks. However, nothing prevents people to cover-up these traces in order to deceive such methods. Understanding the boundaries of forensic analysis is crucial so that efforts are made to surpass them. In the article “Camera Trace Erasing", Chen et al. propose a Siamese Trace Erasing method aiming at extracting those traces. In order to do so, the authors design a novel hybrid loss for network training. This hybrid loss defined as a combination of three different losses: the embedded similarity loss, the truncated fidelity loss and the cross-identity loss. In this section we briefly explore the method and its results.

This work is accepted for publication as *SiamTE: Siamese Trace Erasing for camera trace extraction* on IPOL. We also offer an online demo for anyone to test their own images: https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000443.

9.1 Introduction

The traces left throughout the image formation process (see Chapter 2) encode information about the camera processing chain. These cues play a crucial role in various forensics tasks, such as forgery detection (see Part I) and source camera characterization (see II). However, nothing prevents people to cover-up these traces in order to deceive methods based on camera traces. Understanding the boundaries of forensic analysis is crucial so that efforts are made to surpass them.

Counter-forensics emerged as the research domain that challenges digital forensics and methodically investigates its limitations [22]. In the article “Camera Trace Erasing", Chen et al. propose a siamese trace erasing method aiming at extracting those traces while preserving the image content. In order to do so, the authors design a novel hybrid loss for network training. This hybrid loss defined as a combination of three different losses: the embedded similarity loss, the truncated fidelity loss and the cross-identity loss. The final goal of such an approach is to reveal the weakness of the trace-based forensic methods.

9.2 Problem formulation

Given an image I, the authors state that we can decompose it in two components: The content signal S and the camera trace T[1]:

$$I = S + T. \tag{9.1}$$

In order to extract the camera trace T, the goal is to find $F : \mathcal{X} \to \mathcal{X}$ such that $F(I) = S$, where $\mathcal{X}$ stands for the space of all images. However, finding such F amongst all possible functions $F : \mathcal{X} \to \mathcal{X}$ is not straight-forward. To do so, the authors define two properties, related to the desired camera trace properties.

Firstly, the camera trace should be a distinctive trace of images. Let $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a similarity function related to camera traces, meaning that the similarity between two images increases as the distinctive part (the camera trace) decreases. Given two camera trace extractors F_1 and F_2, we say F_1 is *better* than F_2 if:

$$\phi(F_1(I_1), F_1(I_2)) > \phi(F_2(I_1), F_2(I_2)), \tag{9.2}$$

for all pair of images (I_1, I_2) having different camera traces. The intuition behind this property is that the better we remove the camera traces of two images captured with different camera types, the worse the similarity function is able to distinguish them.

Secondly, we want the camera trace to be the only distinctive part. Given a pair of images (I_1, I_2) having different camera traces and a camera trace extractor F, if we define $S_i^F = F(I_i)$ and $T_i^F = I_i - F(I_i)$ for $i = 1, 2$, then the ideal camera trace extractor F should satisfy:

$$\phi(S_1^F + T_2^F, S_2^F + T_1^F) = \phi(S_1^F + T_1^F, S_2^F + T_2^F). \tag{9.3}$$

This condition implies not only that the camera trace erasing method effectively removes the camera traces but also that it does not introduce a new trace that increases the similarity.

9.3 Method

9.3.1 Architecture

The proposed camera trace erasing method is a parametric function F_θ were θ are trainable parameters. The architecture used by the authors is based on the DDFN presented in [30] and depicted in Figure 9.1. It consists of three modules, a feature extractor, a feature integrator and a reconstruction block. This architecture adopts dense connections, dilated and classical convolution layers and path-widening branches.

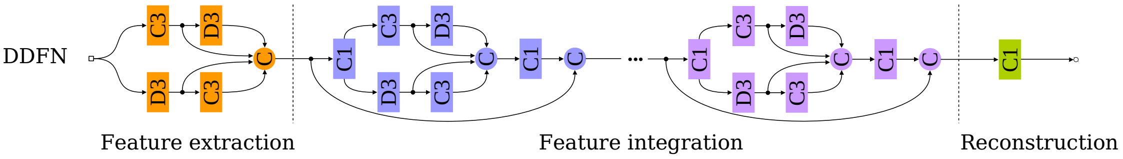


Figure 9.1: Figure extracted from Figure 2 of the article [30]. Rectangular blocks denote convolutions, being "C" the classical convolution and "D" its dilated variant. The following "1" and "3" denote the kernel size. Circular blocks with the "C" letter denote concatenation. Each layer in DDFN (except the last one) uses ReLU as activation function.

[1] It is not clear from this formulation if random noise not related to the camera is included in S or in T.

It is important to mention that the architecture design is not the main focus of this work. Indeed, the authors state that the embodiment of F_θ could be addressed by other network structures.

9.3.2 Proposed hybrid loss

In order to train the model, the authors adopt a siamese configuration with weight sharing. Figure 9.2 depicts such a configuration in the case of two branches. However, the method can be generalized to process an arbitrary number of images by adding more branches. The authors also come up with a novel loss design, an hybrid loss consisting of three different losses: the embedding similarity loss, the cross-identity loss and the truncated fidelity loss. The flowchart of such losses is also depicted in Figure 9.2.

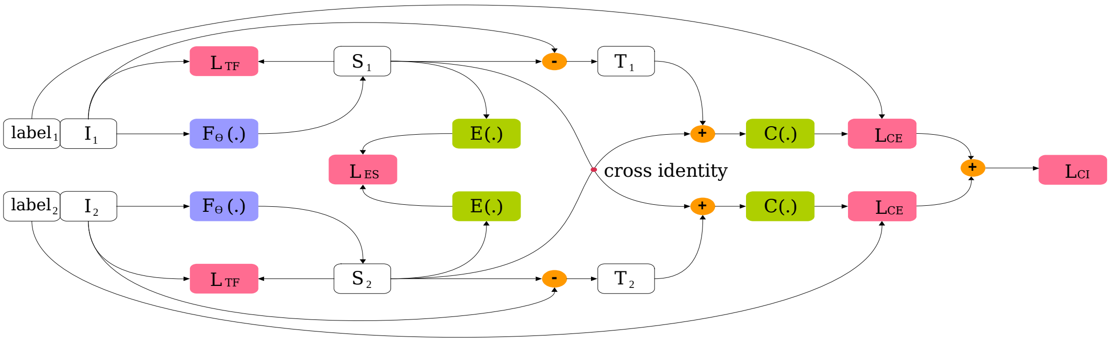


Figure 9.2: Figure based on Figure 3 of the article [29]. The figure depicts a simplified version of the configuration used for training, for the case of two input images processed at the same time. First, the two images having different labels are fed to the network. The content signal S is extracted from each of these images. The signal contents go through a fixed embedding after which the embedding similarity loss is computed. Together with the original image I, the extracted signals are used to compute the truncated fidelity loss. Finally, the camera traces of each of the input images are extracted and plugged in the other one. These synthetic images go through a fixed classification network whose outputs are used, together with the input labels, to compute the cross-entropy loss, which is then added to the corss-identity loss.

- **Embedded similarity loss (L_{ES}).**

  Inspired by Equation 9.2, the authors propose the embedded similarity loss (L_{es}). The computation of such loss consist in, firstly, applying a fixed embedding E in a trace-related space to the content signals extracted in a fixed trace-related space. In this embedding space, the features are firstly normalized and then used to compute the euclidean distance between them, as a similarity score.

- **Cross identity loss (L_{CI}).**

  Inspired by the property stated in Equation 9.3, the authors propose the cross-identity loss. Suppose we have a group of G images coming from different devices. Let $\{I_g\}_{g=1}^G$ be, together with its corresponding labels $\{1, \ldots, G\}$, the input of the network during the training phase, and let $S_g = F_\theta(I_g)$ and $T_g = I_g - S_g$ the extracted signal content and camera trace for image I_g, for all $g = 1, \ldots, G$. For each g, the authors construct the synthetic images

$$I_i^g = S_i + T_g \text{ for all } i = 1, \ldots, G \text{ such that } i \neq g, \tag{9.4}$$

by plugging the traces of device g into the signal content extracted from each of the other devices.

The goal of the cross-identity loss is to maximize the probability of I_i^g of being captured by device g, for all $i = 1, \ldots, G$ such that $i \neq g$.

- **Truncated fidelity loss (L_{TF}).**

  The two previous losses focus on extracting the camera traces while avoiding the introduction of new ones. Still, none of them guarantees the preservation of the signal content. A fidelity loss is needed in order to keep the output image similar to the input one. However, the output image should not be exactly similar to the original one, since the camera traces are to be removed. Still, the camera trace is a weak signal with respect to the content signal. By introducing a margin T, the authors define the truncated fidelity loss:

$$L_{\text{TF}}(I) = \begin{cases} |I - F_\theta(I)| & \text{if } |I - F_\theta(I)| > T \\ 0 & \text{if } |I - F_\theta(I)| \leq T \end{cases}, \tag{9.5}$$

  where $|\cdot|$ stands for the Manhattan (L_1) distance. An appropriate threshold T, allows to preserve the essential manipulation required to extract the camera trace while avoiding the potential over-manipulation in the image content.

Finally, these three losses are linearly combined to build the hybrid loss:

$$L_{\text{H}} = \lambda_1 L_{\text{ES}} + \lambda_2 L_{\text{TF}} + \lambda_3 L_{\text{CI}}, \tag{9.6}$$

where λ_1, λ_2 and λ_3 are the weighting factors.

9.3.3 Implementation details

In this section we provide the implementation details of the embedded similarity loss (Algrithm 14) and the cross-identity loss (Algorithm 15), as given by the authors. It is important to mention that the training code has not been released and, therefore, it is impossible to us to verify the correspondence between the given pseudo-codes and the actually implemented code.

In both implementations the authors use a shift operator to speed up computations. Indeed, this shift operators enables one-to-one operations to be computed in parallel. As a consequence, the number of sequential operations is reduced from ${}_GP_2$ (for non-commutative operations) or ${}_GC_2$ (for commutative operations) to $G - 1$, where ${}_GP_2$ denotes G permute 2 and ${}_GC_2$ stands for G choose 2.

Algorithm 14: Computation of the embedded similarity loss

```
Input: ${I_g}_{g=1}^{G}$ : a group of $G$ images with different camera traces;
Input: $F_\theta(\cdot)$ : a camera trace erasing function with parameters $\theta$;
Input: $E(\cdot)$ : a fixed image embedding;
Input: $N(\cdot)$ : $L_2$ normalization operator;
Input: $\text{shift}_k(\cdot)$ : a shift operator with step $k$;
Input: $D(\cdot)$ : the euclidean distance;
Input: $M$ : a margin for the euclidean distance.
Output: Embedded similarity loss of ${I_g}_{g=1}^{G}$

Initialize feat as en empty vector    # vector to store the embedded content signals
for $g \in \{1, \dots, G\}$ do
    sig $\leftarrow F_\theta(I_g)$    # extract the content signal from $I_g$
    $\text{feat}_g \leftarrow E(\text{sig})$    # embed the content signal using the embedding $E(\cdot)$
    $\text{feat}_g \leftarrow N(\text{feat}_g)$    # normalize the resulting features
    feat $\leftarrow [\text{feat}, \text{feat}_g]$    # append the result to feat
Initialize $L_{\text{ES}}$ to zero
for $k \in \{1, \dots, G-1\}$ do
    dist $\leftarrow \max(0, D(\text{feat}, \text{shift}_k(\text{feat})) - M)$    # distance between features up to a margin
    $L_{\text{ES}} \leftarrow L_{\text{ES}} + \text{mean}(\text{dist})$
$L_{\text{ES}} \leftarrow L_{\text{ES}}/(G-1)$
return $L_{ES}$
```

Algorithm 15: Computation of the cross-identity loss

```
Input: $\{I_g\}_{g=1}^{G}$ : a group of $G$ images with different camera traces;
Input: $\{\ell_g\}_{g=1}^{G}$ : labels corresponding to $\{I_g\}_{g=1}^{G}$;
Input: $F_\theta(\cdot)$ : a camera trace erasing function with parameters $\theta$;
Input: $C(\cdot)$ : an image origin classifier;
Input: $\text{shift}_k(\cdot)$ : a shift operator with step $k$;
Input: $L_{\text{CE}}$ : operator to compute the cross-entropy loss.
Output: Cross-identity loss of $\{I_g\}_{g=1}^{G}$
Initialize sig as en empty vector      # vector to store the extracted content signals
Initialize trs as en empty vector      # vector to store the extracted camera traces
for $g \in \{1, \dots, G\}$ do
    $\texttt{sig}_g \leftarrow F_\theta(I_g)$      # extract the content signal from $I_g$
    $\texttt{sig} \leftarrow [\texttt{sig}, \texttt{sig}_g]$      # append the result to sig
    $\texttt{trs}_g \leftarrow I_g - F_\theta(I_g)$      # extract the camera traces from $I_g$
    $\texttt{trs} \leftarrow [\texttt{trs}, \texttt{trs}_g]$      # append the result to trs
Initialize $L_{\text{CI}}$ to zero
for $k \in \{1, \dots, G-1\}$ do
    $\texttt{pred} \leftarrow C(\texttt{sig}) + \text{shift}_k(\texttt{trs})$      # obtain feedback from $C$ on synthetic images (Equation 9.4)
    $\texttt{loss} \leftarrow L_{\text{CE}}(\texttt{pred}, \text{shift}_k(\{\ell_g\}_{g=1}^{G}))$      # cross entropy loss of pred with shifted labels
    $L_{\text{CI}} \leftarrow L_{\text{CI}} + \text{mean}(\texttt{loss})$
$L_{\text{CI}} \leftarrow L_{\text{CI}}/(G-1)$
return $L_{CI}$
```

9.3.4 Training settings

Training is performed in an extended version of the Kaggle Camera Model Identification (KCMI) dataset. It contains 2,750 images coming from 10 different camera models: Sony NEX-7, Motorola Moto X, Motorola Nexus 6, Motorola DROID MAXX, LG Nexus 5x, Apple iPhone 6, Apple iPhone 4s, HTC One M7, Samsung Galaxy S4 and Samsung Galaxy Note 3. The authors first separate 550 images from this dataset, that are then used for evaluation. Therefore, only 2,200 from the KCMI dataset are left for training and validation. To supplement the training dataset, the authors downloaded from Flickr 2,800 extra images coming from the same 10 camera models. We shall call this dataset KCMI+. Note that, in KCMI+, each camera model has 500 representative images.

The image origin classifier $C(\cdot)$ used to compute L_{CI} as detailed in Algorithm 15 is a ResNet trained on KCMI+. The weights in this ResNet are initialized using an ImageNet pretrained model [35]. Once the image origin classifier is trained, the stacked convolutions in this network are used as the embedding function $E(\cdot)$ for the calculation of L_{ES}, as specified in Algorithm 14.

The camera trace erasing method is also trained on KCMI+ dataset. Images from KCMI+ are randomly copped into patches of size 336×336. The network is fed with groups of patches of size G, which is set to 4. These groups are made up randomly but must satisfy that the four patches come from different types of camera. The mini-batches used for gradient descent are formed with 64 groups of patches random chosen. Adam is used as the optimization algorithm for training with momentum factor set to 0.9. The ratio between the weighting factors in the definition of the hybrid loss (Equation 9.6) $\lambda_1 : \lambda_2 : \lambda_3$ is set to $3 : 1000 : 1$ or $3 : 500 : 1$. As for the hyper-parameters, the margin M in the computation of L_{ES} in Algorithm 14 is set to 0.5 and the truncated fidelity threshold T in Equation 9.5 to 3.

9.3.5 The role of each loss

The authors conduct an ablation study to analyse the role of each of the components of the hybrid loss. Three aspects are evaluated on the conducted ablation study: the effectiveness of the camera trace erasing method (i.e. how well the camera trace is erased), how natural the output image looks and the similarity between the original image and output one. To evaluate the effectiveness of the method, the authors compute the accuracy on the image origin classification task after erasing the camera traces. To do so, they use the two source camera classifiers presented in [125], which obtained the second place in the Camera Model Identification Challenge hosted by the IEEE Signal Processing Society[2]. To evaluate the quality of the images once the camera trace is erased the authors use the NIQE [169] while, to evaluate the similarity to the original image, they use the Manhattan (L_1) distance. Here, instead of reproducing the quantitative results obtained, we will present them qualitatively[3].

- **The role of the embedded similarity loss (L_{ES})**

  Removing L_{ES} has no effect on the image quality nor the Manhattan distance to the original image. However, the effectiveness is highly reduced when this loss is not taken into account. Indeed, the accuracy of the camera origin classifiers is doubled, with respect to the baseline, when removing L_{ES}. Minimizing the cross-identity loss requires the removal of the camera traces up to some extent but it seems that partially removing the camera traces is enough to minimize L_{CI}. Therefore, this loss -together with the truncated fidelity loss- are not enough to guarantee the effectiveness of the method.

- **The role of the truncated fidelity loss (L_{TF})**

  Removing L_{TF} has a positive effect on the effectiveness of the method. Indeed, the accuracy of the classifiers is halved with respect to the baseline when this loss is not taken into account. However, the resulting image quality is deteriorated: the NIQE value grows as well as the Manhattan distance to the original image. This makes sense since the motivation to introduce this loss is avoiding the over-manipulation of the image content.

- **The role of the cross-identity loss (L_{CI})**

  Removing L_{CI} has a positive effect on the effectiveness of the method while keeping the Manhattan distance to the original image similar to the baseline. However, the NIQE value gets worse. The truncated fidelity loss together with the embedded similarity loss are not enough to guarantee a visually natural result. Indeed, the authors show that removing the cross-identity loss generates unpleasant visual artifacts.

[2] https://www.kaggle.com/c/sp-society-camera-model-identification

[3] Quantitative results can be found in Table 4 of the paper [29].

9.4 Demo

The demo takes an input an image with, at most, 1000×1000 pixels. This restriction is imposed for the method to run on real-time. However, interested users can download the source code and run it on images of any size. The output of the demo is the camera-traces-erased version of the image as well as the residual image. The Manhattan distance between the input and the output is provided, as well as the NIQE of both, the input (for reference) and the output. The displayed Manhattan distance is computed as an average rather than as a sum to avoid any size bias and the implementation of NIQE is the one available in the Pytorch toolbox for image quality assessment [28].

9.5 Experiments

In this section we conduct some experiments on the described method. Firstly, we conduct an inspection on the quality of the results in some particular cases. Secondly, we analyse its effectiveness indirectly by conducting a source camera model comparison [71, 162] on the results of the method. Finally, we attempt a direct evaluation of the effectiveness focusing on a particular camera trace: the JPEG artifacts [176].

9.5.1 Quality inspection

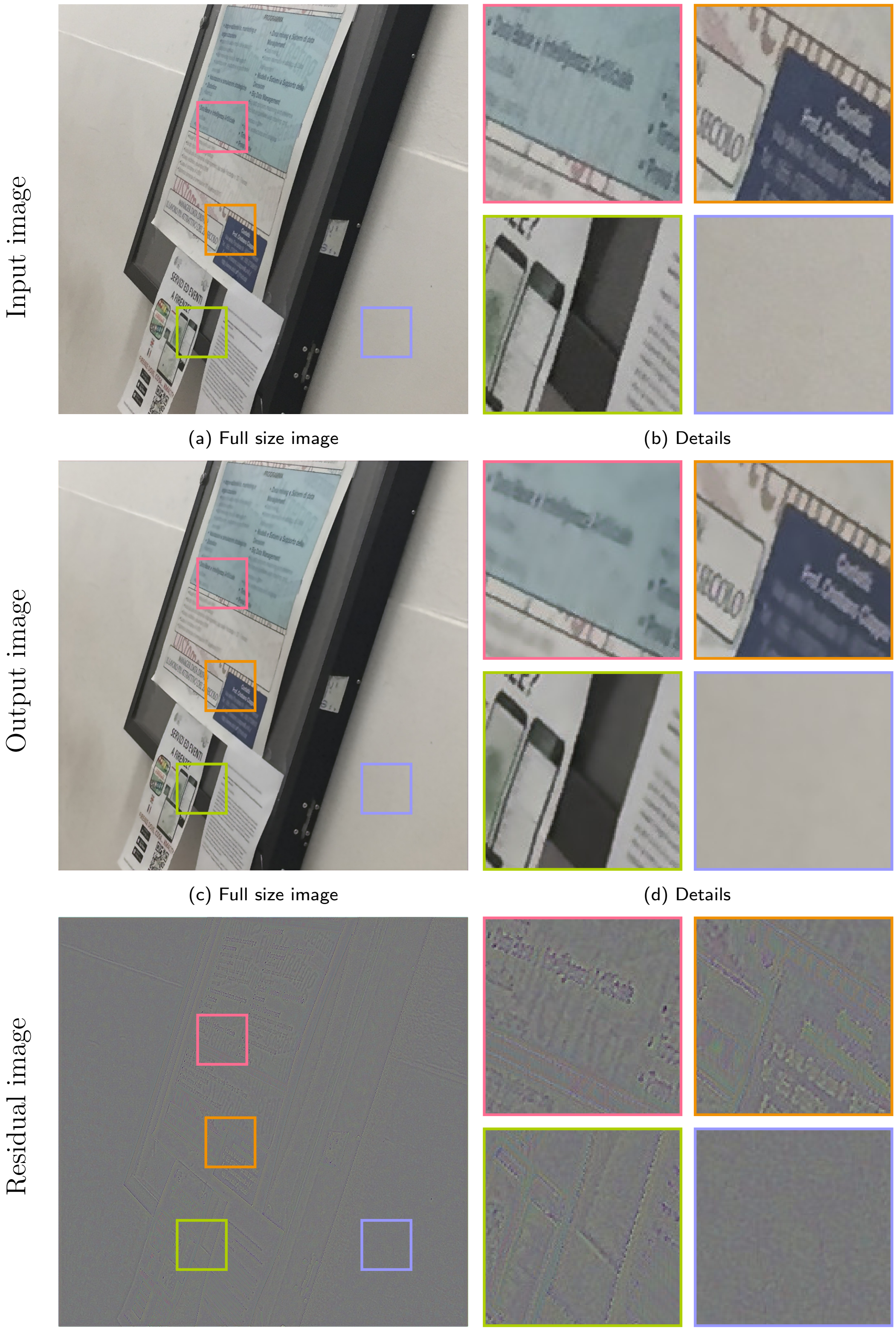
The goal of this experiment is to evaluate the quality of the output images once the camera traces are erased. Figure 9.3 and Figure 9.4 present the results obtained on images `D15_I_nat_0070` and `D19_I_nat_0122` from VISION dataset [193]. The first row of each figure shows the input images, the second row the output images and the third row the residual images. Each row displays both, the full size image (left) as well as some zoom-in details (right).

Firstly, we observe that there is a content leakage in the residuals. Indeed, edges and textures present in the input image are leaked to the residual. This causes a loss of sharpness in the output images. This loss of details in the output image can even lead to the complete destruction of structures present in the original image, as the orange patch in Figure 9.4 shows. On the other hand, the residual on flat patches seems not to present any fixed pattern, as shown on the blue patch in Figure 9.3 and on the blue and green patches in Figure 9.4. However, it is interesting noting that the variance in the blue and green patches in Figure 9.4 are different, probably due to the fact that the original patches present different intensity levels.

The Manhattan distance to the original is approximately 1.5 in both, Figure 9.3 and Figure 9.4. This means that, in average, the absolute difference between the input and output images at a pixel level is 1.5. Regarding the NIQE, we observe a deterioration in its value with respect to the original image in both examples. Though the actual NIQEs observed in the two examples are quite different, in both cases this deterioration is expressed as an increase on the NIQE value by a factor of 1.4.

9.5.2 Indirect effectiveness analysis

The goal of this experiment is to illustrate the effectiveness of the method by performing source camera model comparisons between the input image and the output image. To do so, we use the forensic similarity approach [71, 162]. The conducted experiments were performed taking 128×128 patches without overlap and performing 100 patch-to-patch comparisons. Results are shown in Figure 9.5 for three different camera models from the VISION dataset [193].



(a) Full size image

(b) Details

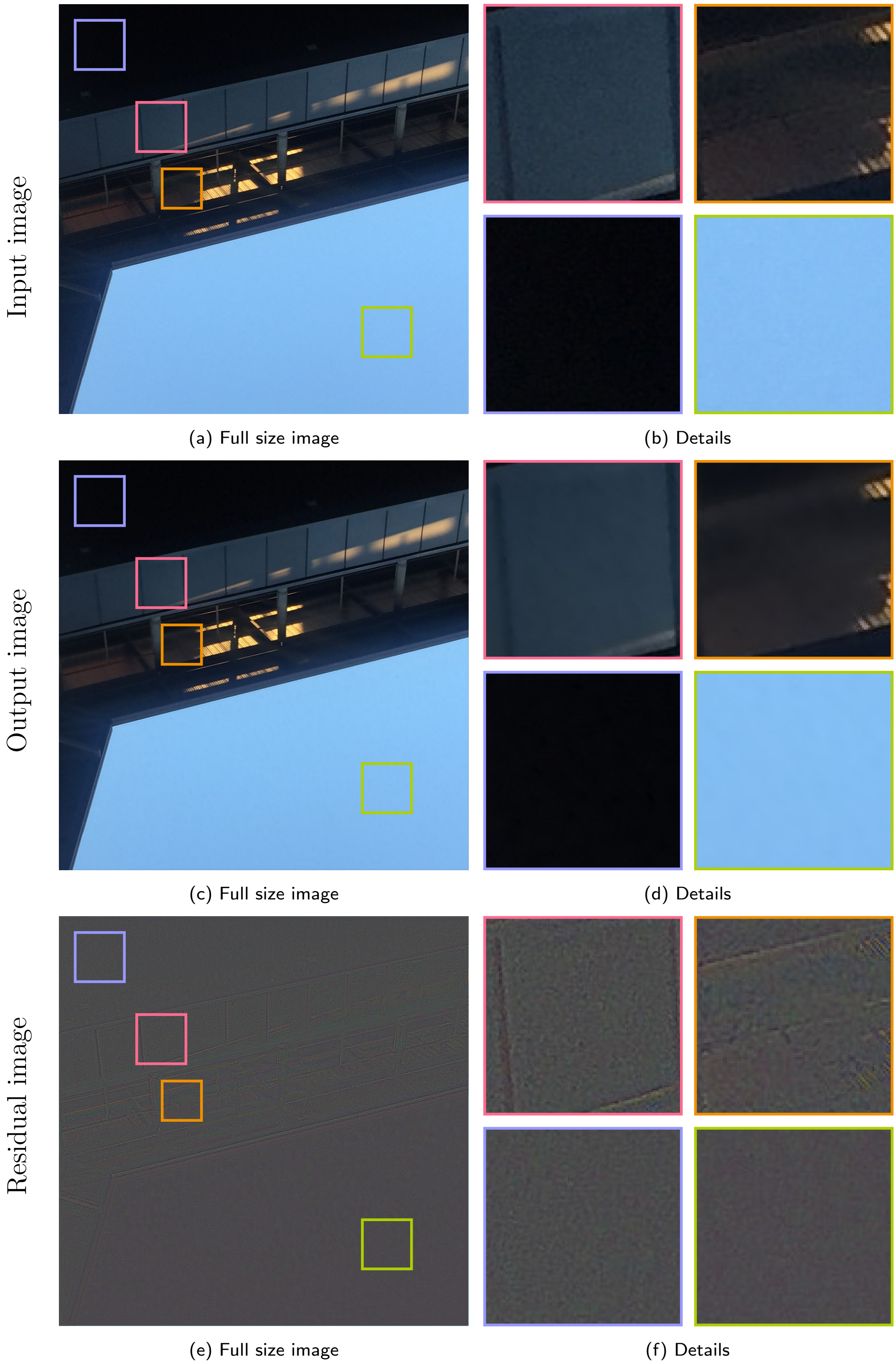
(c) Full size image

(d) Details

(e) Full size image

(f) Details

Figure 9.3: Results on image D15_I_nat_0070 from VISION dataset [193]. The Manhattan distance between input and output is 1.5198, the NIQE of the input is 3.9870 and the NIQE of the output is 5.6754.



(a) Full size image

(b) Details

(c) Full size image

(d) Details

(e) Full size image

(f) Details

Figure 9.4: Results on image D19_I_nat_0122 from VISION dataset [193]. The Manhattan distance between input and output is 1.5080, the NIQE of the input is 6.2907 and the NIQE of the outputis 8.9289.

The first column displays the original image, which is taken as the reference image and also as image 1 in the comparison. The middle column displays the output image, taken as image 2. Finally, the third column shows the histogram of the forensic similarity scores obtained in the 100 patch-to-patch comparisons.

We observe that in all cases the method is able to deteriorate the results of the source camera comparison. The original image, when compared to itself, exhibits similarity scores that concentrate around 1. This concentration is attenuated in all cases when the original image is compared to the one having its camera traces erased. However, this new comparison does not lead to similarity scores concentrating around 0 as we should expect for images having different forensic traces. In all cases, though the similarity scores are lowered, they are still mostly above 0.5.

9.5.3 Effectiveness analysis on JPEG traces

Direct evaluation of the effectiveness of the camera trace erasing method would require a complete model for the in-camera processing process and the design of specific tools to extract each of the traces related to it. Though modelling the complete pipeline can be difficult, there are some specific traces that can be directly analyzed. This is the case of JPEG traces, for which specific methods to detect JPEG artifacts with high confidence are available [176]. In this experiment, we will illustrate the effect of the camera traces erasing method on the presence of JPEG grids.

To do so, we take as a departure point an uncompressed image and compress it at different qualities: 99, 97, 95 and 90, using ImageMagick [101] `convert` with the `quality` flag indicating the corresponding quality factor. the resulting images then go through the camera erasing method. Finally, we compare the grid detection on the original images to the detection on the processed ones. The results of this experiment on image `kodim07` from the Kodak Lossless True Color Image Suite dataset [182] are depicted in Figure 9.6, where the second and fourth column display the JPEG grid detection results.

We observe that for high quality JPEG compression ($Q = 99$), the method effectively erases the JPEG traces. For the rest of the cases ($Q = 97, Q = 95$ and $Q = 90$), ZERO [176] still detects the main grid even after erasing the camera traces. However, the confidence of such detections is lowered as it can be seen from the $log(NFA)$ values displayed. Even in these failure cases, the traces are removed in most regions of the image, as it can be observed from the votemaps. Indeed, the removal of the traces is not homogeneous along the image: some textured zones seem to better preserve the JPEG artefacts.

9.6 Conclusion

In this section, we briefly described the CamTE camera trace erasing method and analyzed its results in different cases. The proposed hybrid loss seems to be an adequate approach to tackle both, the extraction of the camera traces and the preservation of the content signal.

However, though the method makes the camera traces less significant, their detection is still possible in some cases such as strong JPEG-compression. Furthermore, though the resulting image delivers good image quality metrics, the CamTE residuals present some content leakage, mainly regarding edges and textures. Indeed, regarding the results presented in Section 9.5.1 and in Section 9.5.3, the method seems to work better on flat regions than in textured ones, for both the quality of the output and the effectiveness of the trace erasing.

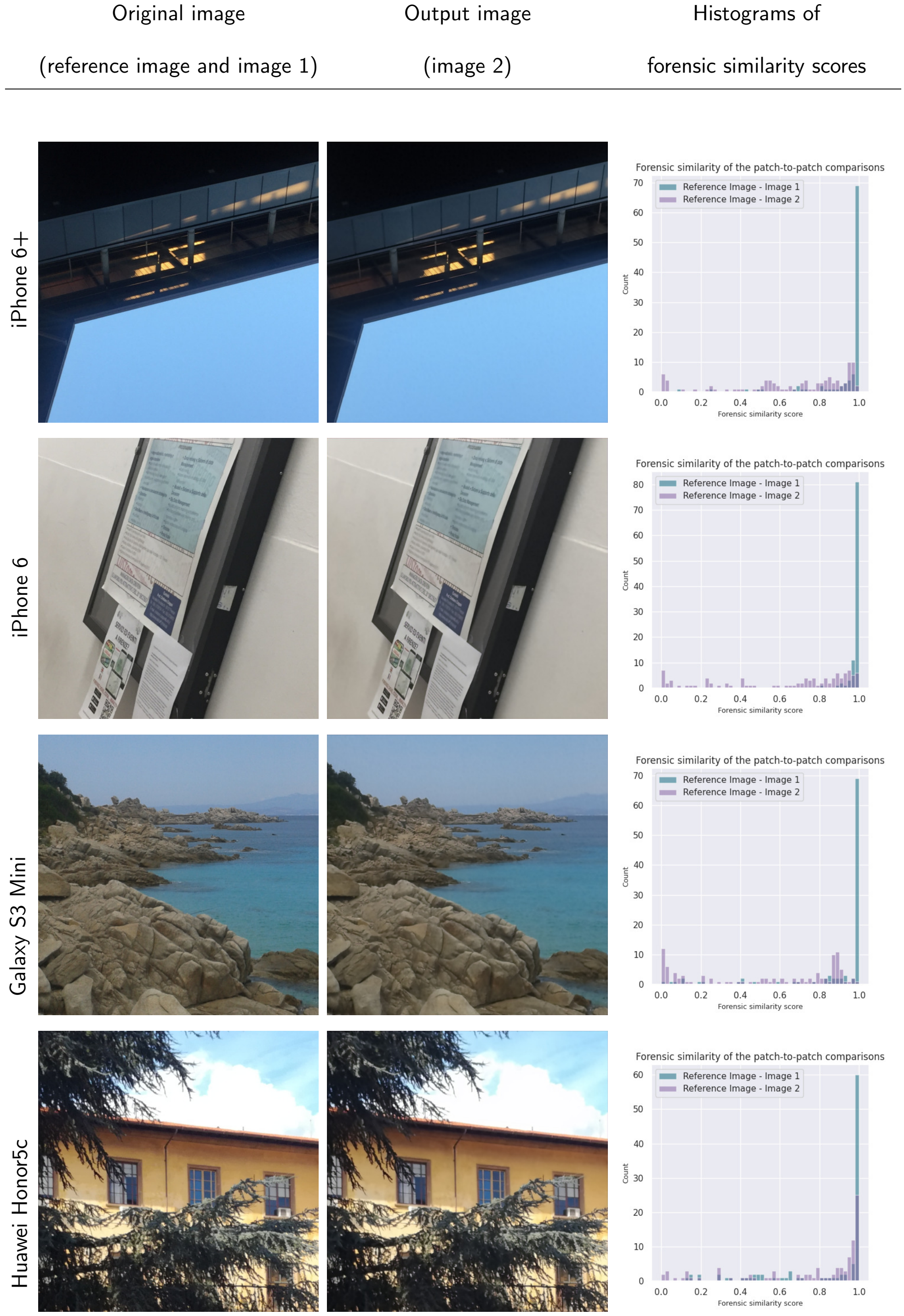


Figure 9.5: Effectiveness of the camera trace erasing method to deceive the forensic similarity approach [71, 162]. The first column displays the original image, which is taken as the reference image and also as image 1 in the comparison. The middle column displays the output image, taken as image 2. Finally, the third column shows the histogram of the forensic similarity scores obtained in the 100 patch-to-patch comparisons.

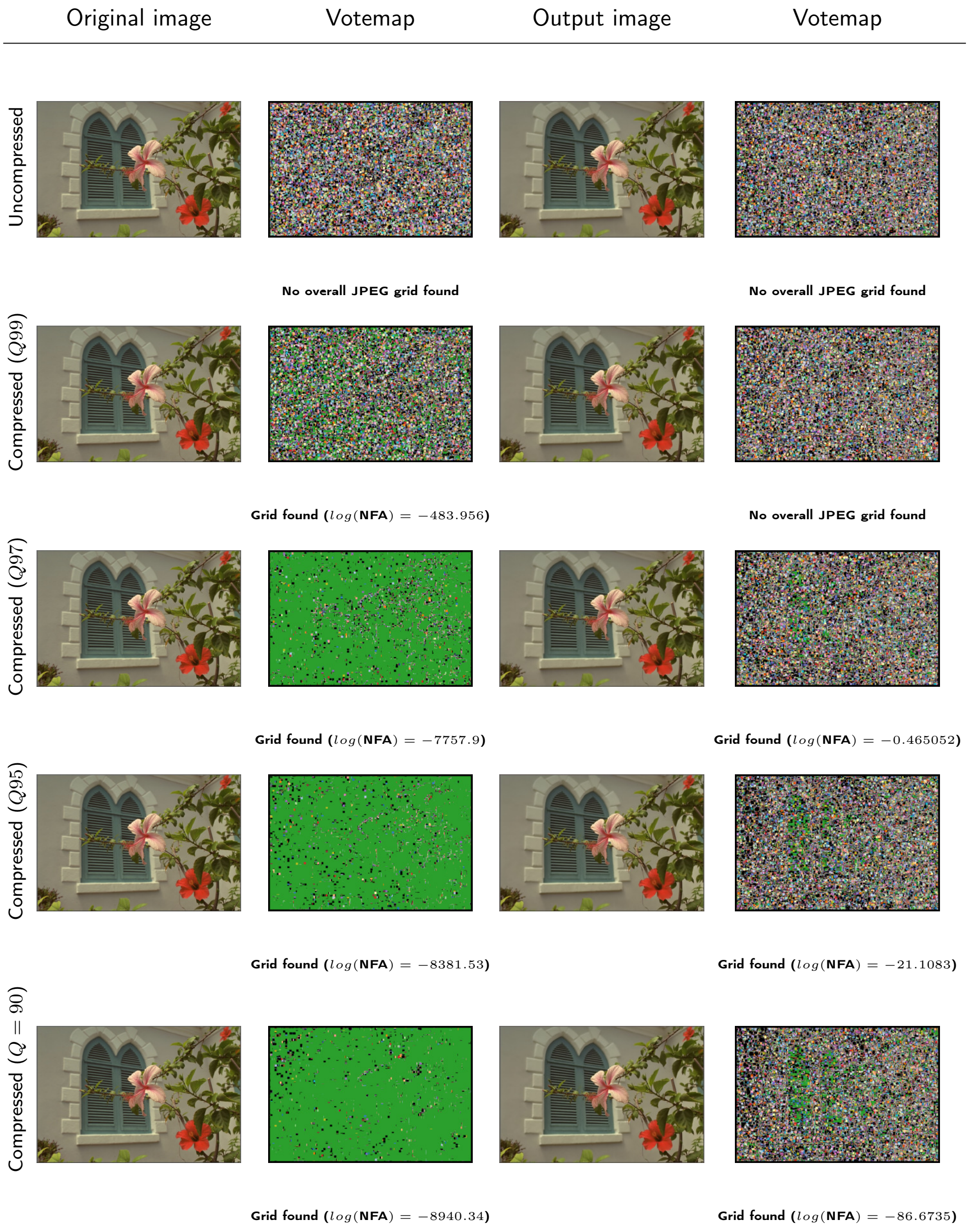

Original image
Votemap
Output image
Votemap
Uncompressed
No overall JPEG grid found
No overall JPEG grid found
Compressed ($Q99$)
Grid found (log(NFA) $= -483.956$)
No overall JPEG grid found
Compressed ($Q97$)
Grid found (log(NFA) $= -7757.9$)
Grid found (log(NFA) $= -0.465052$)
Compressed ($Q95$)
Grid found (log(NFA) $= -8381.53$)
Grid found (log(NFA) $= -21.1083$)
Compressed ($Q = 90$)
Grid found (log(NFA) $= -8940.34$)
Grid found (log(NFA) $= -86.6735$)

Figure 9.6: Effectiveness of the camera trace erasing method to erase the JPEG traces, detected using ZERO [176]. The first column displays the original image and the second column its corresponding JPEG grid detection results. The third column displays the image after erasing its camera traces and the fourth column its corresponding JPEG grid detection results.

Esta página ha sido intencionalmente dejada en blanco.

Chapter 10

Diffusion Models for Image Counter-Forensics

In this chapter, we assess the capabilities of diffusion models to erase the traces left by forgers and, therefore, deceive forensics methods. Such an approach has been recently introduced for adversarial purification, achieving significant performance. We show that diffusion purification methods are well suited for counter-forensics tasks. Such approaches outperform already existing counter-forensics techniques both in deceiving forensics methods, and in preserving the natural look of the purified images.

This work is published as *Diffusion models meet image counter-forensics* in the WACV conference [203].

10.1 Introduction

Image forgeries are present everywhere [64], from fake news on social media [187] to scientific misconduct. Indeed, many image processing tools are available to create visually realistic image alterations. Yet, these modifications leave traces on the image that are tampering cues. Image forensics aims at detecting these alterations by finding local inconsistencies [64]. Image counter-forensics emerged as the research field that challenges forensics methods and explores their limitations [22].

Adversarial attacks share some common properties with image forgeries, in the sense that both techniques introduce subtle modifications to the images that, though imperceptible to the naked eye, disrupt the image's traces. The goal of adversarial attacks is to deceive a model into making incorrect predictions. Adversarial purification can be, therefore, linked to counter-forensics since it aims at prepossessing the input data to remove these adversarial perturbations. Generally, these purification methods are based on generative models [191].

In recent years, diffusion models have emerged as highly effective generative models [96, 197]. These models have showcased impressive capabilities in generating high-quality samples, outperforming traditional Generative Adversarial Networks (GANs) in the realm of image generation. The advancements in diffusion models have led to significant improvements in the fidelity and realism of synthesized images, highlighting their potential as state-of-the-art models in the field.

In this chapter we evaluate, for the first time, the efficiency of diffusion purification methods, currently used for adversarial purification [175, 212], as counter-forensics methods. The rationale behind the use of diffusion models for adversarial purification is that these models

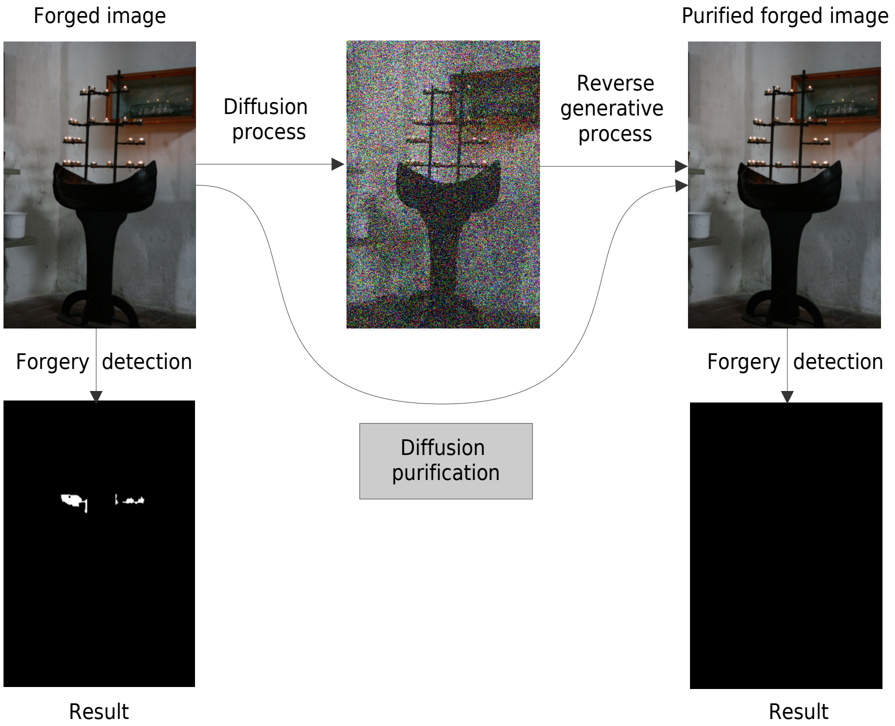


Figure 10.1: Illustration of the use of diffusion models as a counter-forensic technique. A forged image from FAU dataset [37], correctly detected by ZERO [176], produces no detection after diffusion purification.

learn the distribution of clean data. Hence, by diffusing an adversarial example and then applying the reverse generative process, the diffusion model gradually removes the adversarial perturbations and reconstructs the underlying clean sample.

The same rationale can be applied to hide the forensic traces caused by tampering. Indeed, since diffusion models are trained on pristine images, diffusion purification methods applied to forged images should recover purified images without any inconsistency in the camera traces. Once such disruptions on the camera processing chain are erased, purified images should be able to deceive any forgery detection method relying on them. Figure 10.1 shows an example of the aforementioned approach: while ZERO [176] correctly detects the original forgery, once diffusion purification is applied, the method is no longer able to detect it.

10.2 Related work

10.2.1 Image counter-forensics

Counter-forensics attacks can be classified into two categories: the first one corresponds those that focus on a specific trace or method, while the second category corresponds to generic attacks that aim at erasing all the forgery traces and should, therefore, be able to deceive any forensic method.

Among methods that target specific forensic detectors or traces, Fan et al. [62] and Comesana et al. [43] propose attacks against histogram based methods, mainly used to detect JPEG compression traces. Kirchner et al. [118] propose hiding resampling traces by removing the periodic variations in the residual signal in the spatial domain. Do et al. [57] design SIFT-specific attacks that are able to deceive copy-move forgery detectors based on such local descriptors.

With the advent of learning-based forgery detectors, counter-forensic attacks specifically designed for such methods have also been proposed. Marra et al. [159] design a counter-forensic scheme on the feature space. Their goal is to restore the features of the pristine image and, by doing so, to cross the decision boundary of the target detector. In the case of perfect knowledge of the target method, this counter-forensic method delivers great results. However, when the target detector is unknown, the results degrade tremendously. Other methods countering specific learning-based detectors with optimum attack which relies on gradient descent solutions have also been explored [32, 86].

With limited knowledge of forensic models, counter-forensics attackers focus more on erasing the traces by generic tools [15]. The median filter is a technique commonly used as an anti-forensics attack [222], in deep convolutional neural network versions [117] or even variational formulations [194]. Though this method can be effective on several traces, it leaves a distinctive streaking artifact that can be retrieved [119, 227]. To compensate for this, techniques to remove such artifacts have been proposed [68, 194].

More recently, Chen et al. [29] proposed erasing camera traces trying not to damage the signal content by adopting a Siamese-based neural network (see Chapter 9 for more details on this method). Cozzolino et al. [50] and Wu et al. [218] use generative adversarial approaches. Baracchi et al. [14] exploit a real camera firmware to perform the manipulation while reproducing the image statistics. This approach can be most efficient at creating real camera traces, and can easily fool camera identification methods into thinking the image was taken with this camera. However, this method is difficult to use, since it requires disassembling a camera to hack its input field.

10.2.2 Diffusion-based adversarial purification

Nie et al. [175] were the first ones to propose the use of the forward and reverse processes of a pre-trained diffusion model for image adversarial purification. Their method –DiffPure– first diffuses adversarial examples with a small amount of noise. Then, the clean image is recovered by means of a reverse generative process. A very similar idea was developed at the same time by Blau et al. [21]. The theoretical fundamentals justifying the performance of such diffusion-based adversarial purification methods are derived in [223].

Wang et al. [212] face the difficult trade-off between choosing a long diffusion time, which guarantees the removal of the adversarial perturbation, and choosing a small one, which guarantees the similarity between the input image and the purified one. They propose to guide the reverse process by the adversarial image. By doing so, the purified image is forced to stay close to the input image.

Wu et al. [219] also guide the reverse process by the adversarial image. However, they propose to sample the initial input from pure Gaussian noise and gradually denoise it. The rationale of their approach is that the diffused image still carries corrupted structures and the reverse process is likely to get stuck in those corrupted structures.

As the field evolves, several applications of these approaches have been developed. In [5] the authors analyze the performance of DiffPure [175] to purify adversarial attacks on the classification of metastatic tissue. In [199] the authors apply the same principle as in [175] but using an extension of diffusion models to the 3D space [152]. Similarly, [220] also shares the grounds of DiffPure [175] but using a waveform-based diffusion model [120] for adversarial audio purification.

Diffusion purification methods have rapidly gained attention in the field. This interest has even led to questioning the evaluation practices of such techniques [133].

10.3 Background

In this section, we provide a brief overview of Denoising Diffusion Models [96, 196, 197] that will be used as a basis for the next section. Recently, denoising diffusion models, alternatively referred to as score-based generative models, have emerged as a powerful approach amongst generative methods. Denoising diffusion models consist of two processes: a forward diffusion process that progressively adds noise to the input, and a reverse generative process that learns to generate data by denoising.

Forward diffusion process. The diffusion process is a Markov process that gradually adds noise to the clean input data. Let T be the number of steps of the diffusion process, $\mathbf{x}_0$ an input image, and $\mathbf{x}_t$ the forward image until step t $(0 \leq t \leq T)$. The diffusion process from clean data $\mathbf{x}_0$ to $\mathbf{x}_T$ is defined as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{10.1}$$

with

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{10.2}$$

where the variances $\beta_1, \ldots, \beta_T$ are predefined small values.

A notable characteristic of the forward process is that there is a closed-form to generate $\mathbf{x_t}$ at any given time step t directly from $\mathbf{x_0}$ [96]. Indeed, let $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, then we can directly sample $\mathbf{x}_t$ as

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha})}\,\epsilon, \text{ where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{10.3}$$

Reverse denoising process. The reverse generative process is a Markov process that gradually eliminates the noise added in the forward process. The reverse process from $\mathbf{x}_T$ to $\mathbf{x}_0$ is given by

$$p_\theta(\mathbf{x}_{0:T-1}|\mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{10.4}$$

with

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 I), \tag{10.5}$$

where the mean $\mu_\theta(\mathbf{x}_t, t)$ is a trainable network and the variances $\sigma_0^2, \ldots, \sigma_T^2$ can either be fixed or learned using a neural network.

10.4 Proposed method

Our goal is to introduce subtle modifications to a forged image, in order to erase the traces left by the tampering process while, at the same time, preserving the semantic content. The approach we propose is based on diffusion purification methods [175, 212, 219]. It consists of two steps: first, we add noise up to a certain time-step $t = t^*$ in the forward diffusion process, and then we gradually remove it following the reverse diffusion process, up to $t = 0$. We refer to this method as *Diffusion Counter-Forensics*, or shortly *Diff-CF*.

The intuition behind this idea is that the probability distributions of the forged and its corresponding clean image are separated in $t = 0$, but by adding noise in the forward process, the boundaries between the distributions get fuzzier and they begin to overlap, more so the higher the value of t^*. Then, starting from a noisy sample that can belong to either probability distribution, the reverse diffusion process, which was trained on pristine images only, generates a purified version of the image, with no forgery traces. See, for instance, Figure 2 in [165].

The value t^* plays a fundamental role. Intuitively, t^* has to be large enough so that the noise added hides the forgery traces, but small enough so that we can preserve the image semantics and structure. If we set the value of t^* too high, the resulting image would deviate too much from the original one. On the other hand, if the value of t^* is too small, we might not be able to erase the forgery traces correctly. This trade-off is studied more in-depth in Section 10.6.1.

With the purpose of being able to use larger values of t^* without deviating too much from the input image, we also analyze the introduction of guidance in the reverse diffusion process. We refer to this variant as *Counter-Forensics Guided Diffusion*, or *Diff-CFG*. More precisely, we propose to guide the reverse process using the forged image itself, as in [212]. In this way, we encourage the network to produce a clean image as close as possible to the forged one, under the assumption that the forgery traces are subtle enough that they are not reconstructed. In the normal reverse diffusion process, at each time step a new image is sampled following Equation 10.5. Instead, for this variant we propose to sample from

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t) - s_t \Sigma \nabla_{x_t} \mathcal{D}(x_t, x_{in}), \sigma_t^2 I), \tag{10.6}$$

where Σ is the variance of x_t, $\mathcal{D}(x_t, x_{in})$ is some similarity measure between x_t and the input image (forged image) x_{in}, and s_t is a scale factor that depends on the time step t. For high values of t, the forgery traces are completely hidden by the added noise, so we can afford to use large values for s_t, without the risk of guiding the process to reconstruct the forged traces. On the other hand, for small values of t the forgery traces are more retained, and therefore we should use smaller values for s_t. Similar to what is proposed in [212, 219], we define s_t to be proportional to the added noise, as

$$s_t = s\,\frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}, \tag{10.7}$$

where s is a hyper-parameter.

For all experiments, we used the following values: $t^* = 40$, $s = 10^6$, and $\mathcal{D} = -$ SSIM [214] as the guidance metric. A detailed discussion on the influence of the hyper-parameters is presented in Section 10.6. In all cases, the images are divided into patches of 256×256 pixels before running the diffusion process. As for the diffusion model, we used a pre-trained class unconditional checkpoint[1].

10.5 Experiments

To assess the performance of the proposed approaches, we compared both the non-guided (*Diff-CF*) and the guided (*Diff-CFG*) variants with the Camera Trace Erasing technique (CamTE) [29] and with BM3D [51, 131]. While comparison with a plain denoiser is not a common practice in the field, we believe that it should be included. Indeed, camera traces are a sort of noise, in the sense they produce variations in the pixel's values that are not related

[1] `https://github.com/openai/guided-diffusion`

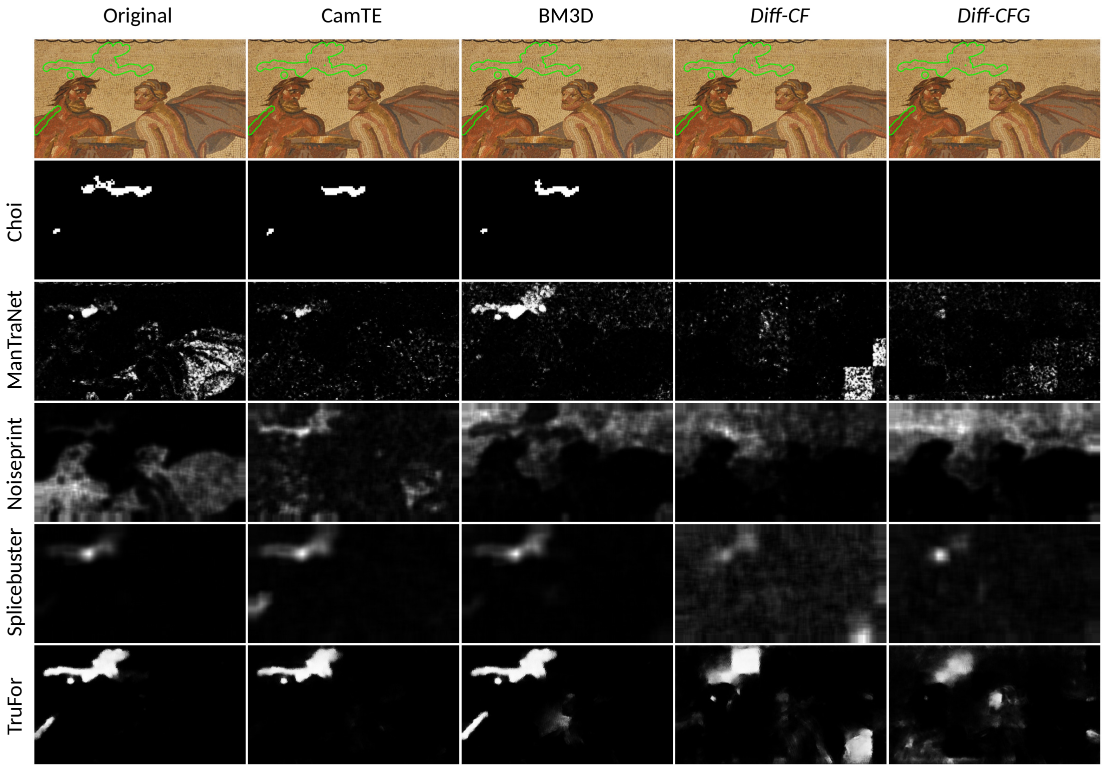


Figure 10.2: Results obtained by different forensics methods on the different versions of image `r7710a7fat` from the Korus dataset [121, 122]. The best two scores are shown in bold and underlined for each database. We observe that Choi [35], ManTraNet [221] and Noiseprint [48] feature no detection when *Diff-CF* or *Diff-CFG* are applied. For Splicebuster [49] and TruFor [90], even if counter-forensics techniques are not completely able to deceive them, the proposed approaches degrade their detections the most.

to the captured scene. On the other hand, we excluded from the comparison the median filtering, which is a popular technique in counter-forensics, since it was shown to be outperformed by CamTE [29].

We ran our comparisons in four image forgery detection benchmark datasets: Korus [121, 122], FAU [37], COVERAGE [215] and DSO-1 [53]. Since most of the methods except for Bammey [10] deliver poor detection results on the DSO dataset, we decided to exclude this dataset from the forgery traces removal analysis and only keep it for image quality assessment.

The goal of counter-forensics methods is to erase all the traces left by the tampering process while preserving the image structures and its semantic content. Therefore, we evaluate two aspects of the counter-forensics techniques under analysis. First, how effectively they hide the forgeries (Section 10.5.1) and second, the quality of the purified images (Section 10.5.2).

10.5.1 Forgery traces removal

The first point to evaluate is how well the proposed approaches remove the forgery traces. To do so, we ran several state-of-the-art forgery detection methods on the original datasets as well as in their counter-forensics versions (images purified using different techniques). To evaluate their capability of deceiving the forensics methods, we look at the difference between the detection performance before and after purification. The forensics methods that were used are: ZERO [176], Noiseprint [48], Splicebuster [49], ManTraNet [221], Choi [11, 35],

		CatNet	Choi	Comprint	MantraNet	Noiseprint	Shin	SpliceBuster	TruFor	ZERO	Noisesniffer	Sim-Graphs	Avg_w
Korus	Original	0.0790 0.0433	0.1971 0.1261	0.0534 0.0461	0.1261 0.0982	0.0988 0.0792	0.0221 0.0568	0.1405 0.1012	0.3428 0.2575	0.0050 0.0028	0.1853 0.1378	0.2452 0.1815	- -
	CamTE	**0.0468** **0.0278**	0.0597 0.0400	0.0356 0.0389	0.0646 0.0644	0.0420 0.0545	0.0305 0.0578	0.0817 0.0729	**0.1961** **0.1489**	0.0000 0.0000	0.1332 0.103	0.1192 0.1113	-0.1024 -0.0479
	BM3D	0.0997 0.0646	0.0352 **0.0227**	0.0278 0.0346	0.0652 0.0746	0.0420 0.0514	0.0155 0.0540	0.0860 0.0744	0.2579 0.1964	0.0000 0.0000	**0.1034** **0.0856**	0.1123 0.0876	-0.0819 -0.0358
	Diff-CF	**0.0418** **0.0204**	**0.0147** 0.0246	**0.0024** **0.0215**	**0.0255** **0.0416**	**0.0190** **0.0360**	0.0027 0.0487	**0.0350** **0.0401**	**0.1454** **0.1131**	0.0045 0.0027	**0.0631** **0.0712**	**0.0484** **0.0534**	**-0.1451** **-0.0677**
	Diff-CFG	0.0852 0.0527	**0.0044** **0.0043**	**0.0125** **0.0281**	**0.0442** **0.0552**	**0.0267** **0.0446**	0.0040 0.0491	**0.0456** **0.0488**	0.2064 0.1601	0.0005 0.0011	0.1060 0.0887	**0.0485** **0.0518**	**-0.1177** **-0.0536**
FAU	Original	0.3228 0.2329	0.3045 0.2670	0.0393 0.0305	0.0203 0.0336	0.0358 0.0482	0.1134 0.1289	0.0074 0.0251	0.4039 0.3373	0.5855 0.5003	-0.03416 0.0161	0.3089 0.2591	- -
	CamTE	**0.0141** **0.0085**	0.1426 0.1173	0.0092 0.0206	0.0154 0.0427	0.0242 0.0434	0.0826 0.1046	0.0045 0.0277	0.0553 **0.0572**	0.0441 0.0288	-0.0460 0.0073	0.0632 0.0803	-0.6120 -0.4259
	BM3D	0.0757 0.0517	0.0679 0.0559	**-0.0017** **0.0126**	-0.0268 0.0377 (0.0041)	**-0.0014** 0.0331	0.0411 0.0803	0.0011 0.0243	0.0802 0.0799	**0.0393** 0.0266	-0.0421 0.0124	0.0103 0.0371	-0.6145 -0.4298
	Diff-CF	**0.0070** **0.0056**	**0.0242** **0.0458**	0.0001 0.0159	0.0057 0.0355 (0.0019)	**-0.0018** **0.0199**	**0.0128** **0.0602**	-0.0050 0.0123	**0.0399** **0.0520**	-0.0007 **0.0015**	-0.0109 0.0429	**0.0000** **0.0263**	**-0.6922** **-0.4687**
	Diff-CFG	0.0241 0.0184	**0.0137** **0.0220**	**-0.0059** **0.0143**	0.0128 0.0339	0.0002 **0.0287**	**0.0202** **0.0646**	0.0127 0.0246	**0.0470** 0.0592	**-0.0043** **0.0008**	-0.0271 0.0295	**0.0020** **0.0208**	**-0.6882** **-0.4688**
COVERAGE	Original	0.2747 0.2199	0.0075 0.0109	0.0230 0.0856	0.2617 0.1856	0.0062 0.0858	0.0615 0.1106	-0.0571 0.0423	0.4442 0.3752	0.0082 0.0070	0.0659 0.0750	-0.0107 0.0569	- -
	CamTE	**0.1480** **0.1162**	0.0056 0.0079	-0.0015 0.0711	0.0790 0.0719	-0.0230 0.0770	**0.0489** **0.1043**	-0.0722 0.0361	**0.2614** **0.2212**	0.0000 0.0000	0.0145 **0.0543**	-0.0341 0.064	**-0.1646** **-0.1048**
	BM3D	0.2666 0.2151	0.0051 0.0036	-0.0281 0.0617	**0.0371** 0.0841	-0.0145 0.0773	0.0515 0.1055	-0.0771 0.0336	0.3267 0.2863	0.0000 0.0000	**-0.0351** **0.0459**	-0.0726 0.0398	-0.1141 -0.0571
	Diff-CF	**0.1598** **0.1278**	0.0011 0.0059	-0.0065 0.0687	**0.0483** **0.0537**	-0.0115 0.0790	0.0514 0.1055	-0.0602 0.0383	0.2849 0.2427	0.0000 0.0000	0.0054 0.0842	-0.0614 0.0489	**-0.1595** **-0.0974**
	Diff-CFG	0.2003 0.1607	-0.0004 0.0010	-0.0124 0.0630	0.0680 **0.0693**	0.0024 0.0858	**0.0475** **0.1051**	-0.0717 0.0334	**0.2738** **0.2386**	0.0000 0.0000	**-0.0098** 0.0664	-0.0477 0.0457	-0.1478 -0.0890

Table 10.1: IoU and MCC results for Korus [121, 122], FAU [37] and COVERAGE [215] datasets and all methods, except for Bammey [10]. For each dataset, we present in the first row the performance of the forgery detectors over the original images. Then, in the following rows, we show the performance of the same detectors over the considered counter-forensic versions of the images, and the difference to the original performance ($\text{metric}_{CF} - \text{metric}_{orig}$). The lower this difference is, the better the counter-forensic method erased the forgery traces. For the sake of readability, methods that are not able to obtain a reasonable performance over the original dataset (MCC < 0.03) are grayed out. Bammey [10] is excluded from this table, as it was not able to obtain an acceptable performance over any of the considered datasets. The last column (Avg_w), is the average of the differences $\text{metric}_{CF} - \text{metric}_{orig}$, weighted by the performance in the original dataset.

Bammey [10], Shin [192], Comprint [158], CAT-Net [126, 127] and TruFor [90].

Choi et al. [35] aims to detect inconsistencies in the mosaic pattern with which the raw image was captured. To do so, they use the fact that sampled pixels were more likely to take extreme values. Also aiming at demosaicing inconsistencies, Shin et al. [192] use the fact that sampled pixels have a higher variance to detect forged regions. Bammey et al. [10] combined the translation invariance of convolutional neural networks with the periodicity of the mosaic pattern to train a self-supervised network into implicitly detecting demosaicing artefacts.

Splicebuster [49] uses the co-occurrences of noise residuals as local features revealing tampered image regions. Noiseprint [48] extends on Splicebuster and uses a Siamese network trained on authentic images to extract the noise residual of an image, which is then analyzed for inconsistencies. TruFor [90] also relies on a noise-sensitive fingerprint that is used with the RGB image to detect deviations from the expected regular pattern that characterizes each pristine image.

Zero [176] targets JPEG artifacts. This method counts the number of null DCT coefficients in all blocks and deduces the grid origin. By doing this locally, this method can detect regions having an inconsistent grid origin. Comprint [158] combines the use of a compression fingerprint with the noise fingerprint in [48]. Comprint is an end-to-end fully convolutional neural network including RGB and DCT streams, aiming at learning compression artifacts on RGB and DCT domains jointly.

ManTraNet [221] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localize forgeries in the image.

To measure detections, we provide scores with the Intersection over Union (IoU) and the Matthews Correlation Coefficient (MCC). In terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), the IoU is the ratio between the number of pixels in the intersection of detected samples and of ground-truth-positive samples and the number of pixels in the union of these sets. Its formula is thus similar to the F_1 score:

$$IoU = \frac{TP}{TP + FN + FP}, \tag{10.8}$$

while the MCC represents the correlation between the ground truth and detections:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{10.9}$$

The scores were computed for each image and then averaged over each dataset. As most surveyed methods do not provide a binary output but a heat map, to adapt the metrics to the continuous setting, we used their weighted version. We regard the value of a heat map H at each pixel u is as the probability of forgery of the pixel. Therefore, given the ground truth mask M, we define the *weighted* TP, *weighted* FP, *weighted* TN and *weighted* FN as:

$$TP_w = \sum_u H(u) \cdot M(u), \tag{10.10}$$

$$FP_w = \sum_x (1 - H(u)) \cdot M(u), \tag{10.11}$$

$$TN_w = \sum_x (1 - H(u)) \cdot (1 - M(u)), \tag{10.12}$$

$$FN_w = \sum_x H(u) \cdot (1 - M(u)). \tag{10.13}$$

IoU and MCC results for all datasets and all methods are presented in Table 10.1. For each dataset, we present in the first row the performance of the forgery detectors over the original images. Then, in the following rows, we show the performance of the same detectors over the considered counter-forensic versions of the images, and the difference to the original performance (metric$_{purified}$ − metric$_{orig}$). The lower this difference is, the better the counter-forensic method erased the forgery traces.

The results in Table 10.1 show that the proposed counter-forensic methods based on diffusion models outperform other counter-forensic techniques in most cases. Indeed, except for the COVERAGE dataset [215] where our methods rank second and third (after CamTE), in all the rest of the datasets *Diff-CF* and *Diff-CFG* achieve the best score reductions. When comparing *Diff-CF* to *Diff-CFG*, we observe that the non-guided version delivers, in most cases, the best results as a counter-forensic method. This can be explained by the fact that, when we do not condition the method, the reverse generative process is able to get closer to the distribution of the clean training data.

Regarding the forensic methods individually, we observe that TruFor [90] outperforms the rest of the methods in most cases. Furthermore, it is the only method that still has an acceptable performance after applying counter-forensics attacks, except on the FAU dataset [37]. Indeed, in this case, once counter-forensic methods are applied, the method delivers highly deteriorated results.

Figure 10.2 shows an example of the results obtained by different forensics methods on the different versions of the same forged image. We observe that Choi delivers pretty much the same result as in the original forgery when CamTE or BM3D are applied. However, it features no detection when *Diff-CF* or *Diff-CFG* are used as counter-forensics techniques. Noiseprint and ManTraNet provide better detections when CamTE or BM3D are applied, respectively. However, no detection is made when using the proposed approaches. On the other hand, none of the counter-forensics methods is able to completely deceive Splicebuster and Trufor. However, we can observe that their results degrade the most when *Diff-CF* and *Diff-CFG* are applied.

10.5.2 Image Quality Assessment

Another important point to evaluate the pertinence of counter-forensic methods is their resulting image quality. We evaluate this quality in two senses. Firstly, we are interested in how natural the purified images are. To evaluate this, we use the reference-free image quality assessment techniques NIQE [169] and BRISQE [168]. Secondly, it is also important to measure the similarity between the input image and the one obtained after the counter-forensic attack. We, of course, want these two images to be perceptually similar. To evaluate this aspect we use the full reference image quality assessment methods LPIPS [237], SSIM [214], and PSNR. For all the metrics we use the implementations provided by the PyIQA library [28].

Results are presented in Table 10.2. For all reference-free metrics, the proposed diffusion-based counter-forensics methods achieve the best performance. For the full-reference metrics, we also obtained the best performance for LPIPS, but BM3D and CamTE get better performance in terms of PSNR and SSIM.

Among the proposed methods, the guided variant always achieves better performance in terms of PSNR and SSIM, as expected. Indeed, the guidance explicitly encourages the purified image to be close to the input image. Still, the results are not so conclusive when evaluating the LPIPS score, where the non-guided version shows a slightly better performance on the Korus dataset.

It is important to mention that, even if *Diff-CFG* uses SSIM as the guidance distance, this does not imply that the obtained scores on that metric should be perfect. In Equation 10.6, the guidance can be interpreted as a sort of gradient descent towards the minimum of $\mathcal{D}(\cdot, x_{in})$. To achieve this minimum, the guidance scale s_t plays a crucial role. Using a non-optimum (in terms of the optimization problem) guidance scale causes the final SSIM score not to be optimal. But this "optimum" guidance scale could not be the best to effectively erase the forgery traces. Section 10.6 studied this trade-off more in-depth.

Regarding the reference-free image quality assessment metrics, *Diff-CF* always achieves better results than *Diff-CFG*. This can be explained by the fact that the unconstrained generative process gets closer to the distribution of the images with which it was trained. Therefore, these images look more natural.

Figure 10.3 shows a qualitative example of the different purified images. We observe that both *Diff-CF* and *Diff-CFG* are good at preserving the fine textures and edges of the image,

		NIQE	BRISQUE	LPIPS	PSNR	SSIM
		(▼)	(▼)	(▼)	(▲)	(▲)
Korus	Original	5.7271	13.7602	0.0000	80.0000	1.0000
	CamTE	5.5442	34.5632	0.1684	**38.2833**	**0.9433**
	BM3D	5.1004	38.0418	0.0835	**43.1409**	**0.9802**
	Diff-CF	**3.8693**	**23.1161**	**0.0733**	32.9680	0.8769
	Diff-CFG	**4.1070**	**28.3290**	**0.0771**	34.3391	0.9126
FAU	Original	4.7392	20.5726	0.0000	80.0000	1.0000
	CamTE	5.8360	40.1577	0.2098	**37.8765**	**0.9460**
	BM3D	5.4875	42.7470	0.1045	**41.2625**	**0.9797**
	Diff-CF	**3.8896**	**19.8268**	**0.0985**	33.0308	0.8792
	Diff-CFG	**4.2440**	**29.9920**	**0.0952**	34.4725	0.9159
COVERAGE	Original	4.5529	19.0256	0.0000	80.0000	1.0000
	CamTE	5.4513	30.3558	0.0631	**35.7974**	**0.9648**
	BM3D	5.8792	35.9560	**0.0237**	**44.1417**	**0.9888**
	Diff-CF	**4.3343**	**17.1298**	0.0281	33.4959	0.9275
	Diff-CFG	**5.0359**	**27.8903**	**0.0276**	34.6969	0.9487
DSO-1	Original	3.9174	16.6183	0.0000	80.0000	1.0000
	CamTE	5.2180	40.2867	0.2022	**38.5459**	**0.9446**
	BM3D	5.1870	39.8485	0.1239	**41.9057**	**0.9750**
	Diff-CFG	**3.3907**	**9.2614**	**0.0950**	34.1204	0.8862
	Diff-CFG	**3.6686**	**19.3601**	**0.0899**	35.3473	0.9154

Table 10.2: Image quality assessment results of the evaluated counter-forensics techniques. The ▼ indicates that the lower the score the better while the ▲ indicates that the higher score the better. The best two scores are shown in bold and underlined for each database. For the no-reference metrics NIQE and BRISQE, the proposed diffusion-based counter-forensics methods achieve the best performance.

while CamTE and BM3D blur all these fine structures. For instance, the details highlighted in the green patch show that the granularity in the cherubs' cheeks is blurred out by BM3D and CamTE, while it is preserved by the diffusion-based models. This is also visible in the cherubs' chin, highlighted in the yellow patch. As for the edges, the sharpness of the nose (green patch) and the lips (yellow patch) are also better preserved by the proposed approaches.

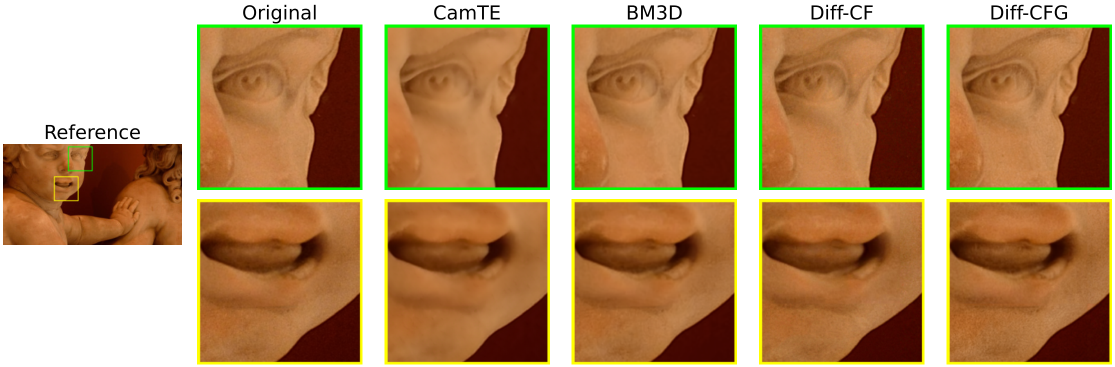


Figure 10.3: Image quality comparison for all considered counter-forensics methods. We observe that both *Diff-CF* and *Diff-CFG* are good at preserving the fine textures and edges of the image while CamTE and BM3D blur all these fine structures.

10.6 Influence of the parameters

The goal of this chapter is to provide a first study on the use of diffusion models as counter-forensics techniques. As such, it is important to evaluate how the results vary along with the parameters. The non-guided approach *Diff-CF* has only one parameter: the time step t^*, while *Diff-CFG* has two: the time step t^* and the guidance scale s. In this experiment, we focus mainly on *Diff-CFG* since we think that the interaction of both parameters is way more complex than analyzing a single one. The experiments in this section are carried out on Korus dataset [121, 122]. We evaluate both: the forgery traces removal capabilities and the image quality of the purified images. For the first, we compute the performance drop for the best performing methods over the original dataset: Choi, MantraNet, Noiseprint, Splicebuster, and TruFor. For the second, we use all the image quality assessment metrics presented in Section 10.5.2.

10.6.1 Diffusion time-step

The results of the impact of the time-step t^* are presented on the left-hand side of Figure 10.4. The analysis is pretty straightforward: the larger the value of t^*, the forgery traces removal performance improves (gets lower). On the other hand, the image quality metrics improve the smaller the value of t^*. There is a clear trade-off in the selection of this parameter, that is simple to understand: with higher values of t^*, we add more noise to the original image in the forward diffusion process, which makes it easier to hide the forgery traces. On the other hand, starting the reverse process too far away from the original image leads to larger deviations between the original image and the purified one.

In addition, it is interesting to note that all the full-reference metrics keep strongly degrading as we increase t^*, but the reference-free metrics seem to follow a more asymptotic behavior. This evidence can be explained due to the fact that, even if the generated images are more apart from the original one, the diffusion process, following the learned distribution, is still generating natural images.

10.6.2 Guidance scale

The guidance scale ensures that the purified image remains close to the manipulated image, thus not modifying its semantic content. However, it is crucial that the chosen guidance scale

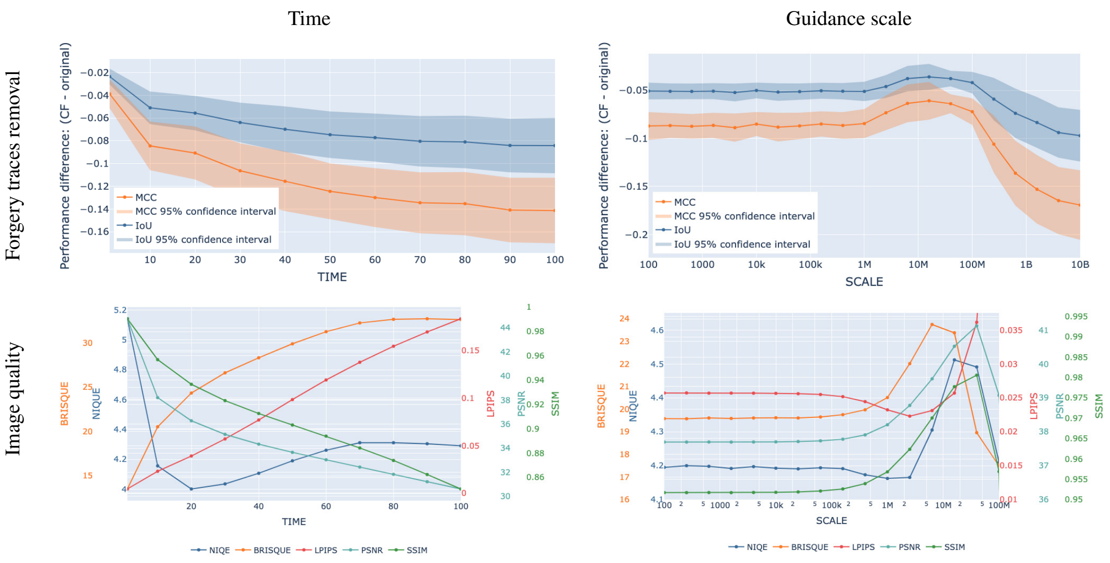


Figure 10.4: Study of the impact of the time-step t^* (left-hand side), and guidance scale s (right-hand side). For each parameter, we evaluate its influence on the forgery traces removal task (top) and on the purified image quality (bottom). For the forgery traces removal task, we plot the average of the difference between the performance before and after purification, for the best performing methods the original dataset, as a function of the parameters' value. The colored background area represents the 95% confidence interval. For the Image quality assessment, all five metrics presented in Section 10.5.2 are plotted as a function of the parameters' value, each one with a different axis, for a better visualization. This figure is best viewed in color. An interactive version of these plots is included in the supplementary material.

is not excessively large since it would cause the purified image to match the adulterated image, potentially retaining the manipulation traces [219].

We conducted a series of experiments to study the scale influence, varying the scale value (s in Equation 10.7), while keeping a fixed time-step $t^* = 10$. As can be seen in the right-hand side of Figure 10.4, the performance difference has small variations for about the first half of the scale range studied, then shows a slight increase, and finally a great drop. The best point we could choose would be with the lowest value, so at first one could be tempted to use the highest value for the scale. But if we add the image quality assessment to the analysis, we observe that for those scale values, the quality of the images is highly degraded. Therefore, an intermediate point should be chosen. Note that the optimal point for the forgery traces removal is not the optimal point in terms of image quality. As mentioned in Section 10.5.2, this could explain why in our experiments, we do not obtain the best performance in terms of SSIM, even though we are guiding the diffusion process with this metric.

10.7 Conclusions and Future Work

In this chapter, we presented a first study on the use of diffusion models for counter-forensics tasks. We showed that such an approach can deliver better results than the already existing techniques for both, forgery trace removal and image quality. Of course, there is a risk that the shown approaches would be used by people wanting to create forgeries and make them look authentic. The simplicity of this method increases this risk. However, it is also because of its simplicity that the method should be made public: It is important to expose the shortcomings of current methods so that one can know how much trust can be put into an image, and so that alternative ways of authentication are developed.

In this direction, future work includes analyzing the traces left by the diffusion purification process [47] to check whether the use of such a counter-forensic approach can be detected or not. Also, it would be interesting to analyze the robustness of the different methods to such kind of counter-forensic methods.

Esta página ha sido intencionalmente dejada en blanco.

Conclusion

This short chapter concludes a thesis dedicated to the study of noise throughout the image formation process and its applications to image forensics. Several contributions were made to the state of the art in image forgery detection and source camera identification. In addition, we also investigated counter-forensics techniques to assess the limitations of various forgery detection methods.

Chapter 1 introduced the subject of image forensics. Here we explored the issues and challenges such subject represents for law enforcement and for journalists, and the tools currently available for those. Furthermore, this chapter summarized the main contents of this thesis and the contributions derived from it.

Chapter 2 reviewed the image processing pipeline and its main operations. We presented the traces left by each of its steps which can be regarded as a sort of image fingerprint embedded during the image formation process. Specifically, we showed how it is reflected in the noise model. In particular, we showed that, after JPEG-compression, the noise present in an image does not fit the additive white Gaussian noise hypothesis but rather it is intensity-dependent and frequency-dependent.

Chapter 3 brings forward a simple, yet effective, method for forgery detection based on noise analysis. Such method computes local noise curves and compares them to the global noise curve estimated from the whole image. By doing so at multiple scales, the method is well-suited to work on JPEG-compressed images. Assuming that forgeries represent a small region of the images and do not modify the global noise curve, regions for which the local noise curve differs from the global one are regarded as suspicious. This method is specially effective on colorization attacks. Indeed, this kind of forgery introduces inconsistencies which only appear when considering intensity-dependent noise models.

Chapter 4 introduces a more sophisticated method than the one in the preceding chapter. This method estimates for each image a background stochastic model which makes it possible to detect local noise anomalies. The algorithm includes an a contrario statistical validation step, which associates a Number of False Alarms (NFA) with each tampering detection. Detections are obtained by a threshold of the NFA, which renders the method fully automatic and endows it with a false alarm control mechanism.

Chapter 5 explored the possibility of learning the camera traces instead of using hand-crafted features. Here we introduced and analyzed the forensic similarity approach by Mayer and Stamm [162] and its use for forgery detection. Such an approach is built on a graph-based representation of images, where image patches are represented as the vertices of the graph, and the edge weights are assigned in order to reflect the forensic similarity between the connected patches. The forensic similarity score is learned through a neural network. In this setting, we showed that forgery detection and localization can be cast as a cluster analysis problem on the similarity graph.

Chapter 6 introduced a novel way to evaluate forgery detection methods. By locally modifying the formation pipeline of an image, we were able to create “non-semantic forgeries”, that contain changes in the underlying traces of the image without changing any of its semantic content. This methodology enables trace-aware evaluation of forensics tools, as it can highlight exactly to which traces each method is sensitive. Noisesniffer (from Chapter 4) beats the rest of the noise-based models and achieves the state-of-the-art performance of learning approaches.

Chapter 7 introduced a new forensic task: source camera model comparison. The goal is to determine if two images were taken with cameras from the same model. The method introduced in this chapter takes over the forensic similarity approach already introduced in Chapter 5 but applies it patch-wise to produce a forensic similarity histogram. Results on this method show that the performance depends on the model, being the results way better whenever the camera models were used during training.

Chapter 8 also focused on source camera forensics but, in this chapter, we introduced PRNU-based methods for source camera identification. Here, we presented a new testing approach to detect the presence of a certain PRNU pattern in a query image. Such an approach relies on two hypothesis tests based on local correlations which do not require computing empirical distributions. Furthermore, the p-value of the test serves as a statistically founded confidence measure that can serve as certification.

Chapter 9 tackled the problem of hiding the camera traces in order to deceive forensics methods relying on those. In this chapter we introduce and analyse the camera trace erasing approach by Chen *et al.* [29]. The goal of such approach is to extract the forensic traces of an image while preserving the image content. To do so, we presented and analysed the novel hybrid loss proposed by the authors which is a combination of the embedded similarity loss, the truncated fidelity loss and the cross-identity loss.

Chapter 10 analysed, for the first time in the literature, the use of diffusion models for counter-forensics tasks. Here, we showed that by diffusing a forged image and then applying the reverse generative process, the camera processing traces regenerated are consistent. Therefore, methods searching for inconsistencies in such cues to detect forgeries become ineffective. Furthermore, the purified images remain semantically close from the input one.

All in all, the main contributions of this thesis are:

- A rigorous theoretical study on the impact of JPEG-compression on prior image noise with experimental validation (Chapter 2).
- The introduction of a new state-of-the-art noise-based forgery detection method, Noisesniffer from Chapter 4.
- The Trace methodology and database, introduced in Chapter 6, which provides a method to estimate the non-semantic detection strengths and weaknesses of all forensic methods.
- A novel approach for source camera authentication that does not require the computation of empirical distributions and delivers, for each test, a p-value that serves as a statistically founded confidence measure.
- A first study on the use of diffusion purification methods for counter-forensics (Chapter 10).

These contributions open up several paths for further exploration. Though Noisesniffer achieves a state-of-the-art performance it looses, by construction, forgeries having higher noise levels than the background image. This choice was made to prevent the method of delivering false alarms due to textures. Further work on how to differentiate texture from noise could be beneficial for the method to extend its scope. Furthermore, a multi-scale approach as the one used in Chapter 3 could also lead to a better performance of the method.

Further work on the Trace database would include a systematic analysis of more camera traces and post-processing applied to the whole image, such as double compression traces. In addition to evaluation, the proposed methodology could also be used to train forgery detection methods. Indeed, it could be used to re-train the forensic similarity approach in Chapter 5 and in Chapter 7 using different artificially generated pipelines instead of camera models.

Regarding PRNU analysis, although the testing approach seems pretty complete, new questions regarding the uniqueness of the PRNU have merged in the literature [102]. Indeed, the PRNU extracted using the classical approaches reviewed in 8 contains traces from the image processing and post-processing pipeline which are specially strong in modern devices. Therefore, such devices trigger an increase in the false alarm rate. Still, we are confident that isolating such non-unique artifacts of finding a region of the spectrum where their magnitude is not so significant can solve this problem.

Counter-forensics seem to be a less explored subject in the literature. Here we presented a first study on the use of diffusion models to cover-up forgeries. This analysis can be extended in several ways. Firstly, by optimizing the parameters to achieve better results. Second, it could also be tested on other forensics tasks rather than forgery detection. Finally, the residual image (i.e. the difference between the input and the purified output) could be used as input for forgery detection since it contains all the relevant forgery traces.

More generally, the last part of this thesis poses a major question on the robustness of trace-based methods. It was shown in the last chapter that simple counter-forensic attacks can significantly decrease the performance of most approaches. Thinking about alternative ways of authenticating images should be in the mind of all the image forensics community.

These brief passages concludes this thesis. I would like to express my gratitude once more to those who supported me throughout this journey, as well as the reader for your interest in my work.

Esta página ha sido intencionalmente dejada en blanco.

Bibliography

[1] C. Aguerrebere, J. Delon, Y. Gousseau, and P. Musé. Study of the digital camera acquisition process and statistical modeling of the sensor raw data. Technical report, Aug. 2013. URL https://hal.archives-ouvertes.fr/hal-00733538.

[2] M. Al-Ani and F. Khelifi. On the spn estimation in image forensics: A systematic empirical evaluation. *IEEE Transactions on Information Forensics and Security*, 12(5): 1067–1081, 2017. doi: 10.1109/TIFS.2016.2640938.

[3] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra. A SIFT-based forensic method for copy-move attack detection and transformation recovery. *IEEE Trans. on Information Forensics and Security*, 6(3):1099–1110, Sep. 2011. doi: 10.1109/TIFS.2011.2129512. URL http://www.lambertoballan.net/downloads/2011_tifs_preprint.pdf.

[4] I. Amerini, R. Becarelli, R. Caldelli, and A. Del Mastio. Splicing forgeries localization through the use of first digit features. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 143–148. IEEE, 2014.

[5] L. L. Ankile, A. Midgley, and S. Weisshaar. Denoising diffusion probabilistic models as a defense against adversarial attacks. *ArXiv*, abs/2301.06871:null, 2023. doi: 10.48550/arXiv.2301.06871.

[6] F. J. Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, 35(3/4):246–254, 1948. ISSN 00063444. URL http://www.jstor.org/stable/2332343.

[7] F. Argenti, G. Torricelli, and L. Alparone. Mmse filtering of generalised signal-dependent noise in spatial and shift-invariant wavelet domains. *Signal Processing*, 86(8):2056–2066, 2006. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2005.10.014. URL https://www.sciencedirect.com/science/article/pii/S0165168405003592.

[8] F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61 3:183–93, 1954.

[9] Q. Bammey. Analysis and Experimentation on the ManTraNet Image Forgery Detector. *Image Processing On Line*, 12:457–468, 2022. https://doi.org/10.5201/ipol.2022.431.

[10] Q. Bammey, R. G. v. Gioi, and J.-M. Morel. An adaptive neural network for unsupervised mosaic consistency analysis in image forensics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Q. Bammey, R. Grompone von Gioi, and J.-M. Morel. Image Forgeries Detection through Mosaic Analysis: the Intermediate Values Algorithm. *Image Processing On Line*, 11: 317–343, 2021. https://doi.org/10.5201/ipol.2021.355.

[12] Q. Bammey, M. Colom, T. Ehret, M. Gardella, R. Grompone, J.-M. Morel, T. Nikoukhah, and D. Perraud. *How to Reconstruct the History of a Digital Image, and of Its Alterations*, chapter 1, pages 1–40. John Wiley & Sons, Ltd, 2022. ISBN 9781119901808. doi: https://doi.org/10.1002/9781119901808.ch1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119901808.ch1.

[13] Q. Bammey, T. Nikoukhah, M. Gardella, R. G. von Gioi, M. Colom, and J.-M. Morel. Non-semantic evaluation of image forensics tools: Methodology and database. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2383–2392, 2022. doi: 10.1109/WACV51458.2022.00244.

[14] D. Baracchi, D. Shullani, M. Iuliani, D. Giani, and A. Piva. Camera obscura: Exploiting in-camera processing for image counter forensics. *Forensic Science International: Digital Investigation*, 38:301213, 2021. ISSN 2666-2817. doi: https://doi.org/10.1016/j.fsidi.2021.301213.

[15] M. Barni, M. C. Stamm, and B. Tondi. Adversarial multimedia forensics: Overview and challenges ahead. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 962–966. IEEE, 2018.

[16] P. Bas, J.-M. Chassery, and B. Macq. Geometrically invariant watermarking using feature points. *IEEE Transactions on Image Processing*, 11(9):1014–1028, 2002. doi: 10.1109/TIP.2002.801587.

[17] B. Bayar and M. C. Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Transactions on Information Forensics and Security*, 13(11):2691–2706, 2018.

[18] F. Bertini, R. Sharma, A. Iannì, and D. Montesi. Profile resolution across multilayer networks through smartphone camera fingerprint. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, IDEAS '15, page 23–32, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334143. doi: 10.1145/2790755.2790765. URL https://doi.org/10.1145/2790755.2790765.

[19] X. Bi, Y. Wei, B. Xiao, and W. Li. Rru-net: The ringed residual u-net for image splicing forgery detection. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 30–39, 2019.

[20] T. Bianchi, A. De Rosa, and A. Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2444–2447. IEEE, 2011.

[21] T. Blau, R. Ganz, B. Kawar, A. Bronstein, and M. Elad. Threat model-agnostic adversarial defense using diffusion models, 2022.

[22] R. Böhme and M. Kirchner. Counter-forensics: Attacking image forensics. In *Digital image forensics*, pages 327–366. Springer, 2013.

[23] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. Tampering detection and localization through clustering of camera-based cnn features. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1855–1864, 2017. doi: 10.1109/CVPRW.2017.232.

[24] B. Brecht. *The Life of Galileo*. Methuen, 1968. Translated by D. I. Vesey.

[25] I. Castillo Camacho and K. Wang. A comprehensive review of deep-learning-based methods for image forensics. *Journal of Imaging*, 7(4), 2021. ISSN 2313-433X. doi: 10.3390/jimaging7040069. URL https://www.mdpi.com/2313-433X/7/4/69.

[26] M. Castro, D. M. Ballesteros, and D. Renza. A dataset of 1050-tampered color and grayscale images (cg-1050). *Data in Brief*, 28:104864, 2020. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2019.104864. URL http://www.sciencedirect.com/science/article/pii/S2352340919312193.

[27] S. Chakraborty. A cnn-based correlation predictor for prnu-based image manipulation localization. *Electronic Imaging*, 2020:78–1, 01 2020. doi: 10.2352/ISSN.2470-1173.2020.4.MWSF-078.

[28] C. Chen and J. Mo. IQA-PyTorch: Pytorch toolbox for image quality assessment. [Online]. Available: https://github.com/chaofengc/IQA-PyTorch, 2022.

[29] C. Chen, Z. Xiong, X. Liu, and F. Wu. Camera trace erasing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, and F. Wu. Real-world image denoising with deep boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(12): 3071–3087, 2020. doi: 10.1109/TPAMI.2019.2921548.

[31] M. Chen, J. Fridrich, M. Goljan, and J. Lukas. Determining image origin and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, 3(1):74–90, 2008. doi: 10.1109/TIFS.2007.916285.

[32] Z. Chen, B. Tondi, X. Li, R. Ni, Y. Zhao, and M. Barni. A gradient-based pixel-domain attack against svm detection of global image manipulations. In *2017 IEEE workshop on information forensics and security (WIFS)*, pages 1–6. IEEE, 2017.

[33] D. Chicco. Ten quick tips for machine learning in computational biology. *BioData mining*, 10:35–35, Dec 2017. ISSN 1756-0381. doi: 10.1186/s13040-017-0155-3. URL https://pubmed.ncbi.nlm.nih.gov/29234465.

[34] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6–6, Jan 2020. ISSN 1471-2164. doi: 10.1186/s12864-019-6413-7. URL https://pubmed.ncbi.nlm.nih.gov/31898477.

[35] C.-H. Choi, J.-H. Choi, and H.-K. Lee. Cfa pattern identification of digital cameras using intermediate value counting. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM&amp;Sec '11, page 21–26, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308069. doi: 10.1145/2037252.2037258. URL https://doi.org/10.1145/2037252.2037258.

[36] K. S. Choi, E. Y. Lam, and K. K. Y. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Opt. Express*, 14(24):11551–11565, Nov 2006. doi: 10.1364/OE.14.011551. URL http://opg.optica.org/oe/abstract.cfm?URI=oe-14-24-11551.

[37] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 7(6):1841–1854, 2012. doi: 10.1109/TIFS.2012.2218597.

[38] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004. doi: 10.1103/PhysRevE.70.066111. URL https://link.aps.org/doi/10.1103/PhysRevE.70.066111.

[39] M. Colom. Noise-free test images dataset. URL http://mcolom.info/pages/no_noise_images/.

[40] M. Colom. *Multiscale noise estimation and removal for digital images*. PhD thesis, Universitat de les Illes Balears, 7 2014.

[41] M. Colom and A. Buades. Analysis and Extension of the Ponomarenko et al. Method, Estimating a Noise Curve from a Single Image. *Image Processing On Line*, 3:173–197, 2013. doi: https://doi.org/10.5201/ipol.2013.45.

[42] M. Colom, M. Lebrun, A. Buades, and J. Morel. Nonparametric multiscale blind estimation of intensity-frequency-dependent noise. *Image Processing, IEEE Transactions on*, 24(10):3162–3175, Oct 2015. ISSN 1057-7149. doi: 10.1109/TIP.2015.2438537.

[43] P. Comesana and F. Perez-Gonzalez. The optimal attack to histogram-based forensic detectors is simple (x). In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 137–142. IEEE, 2014.

[44] A. Cooper. Improved photo response non-uniformity (prnu) based source camera identification. *Forensic science international*, 226 1-3:132–41, 2013.

[45] S. Corchs, F. Gasparini, and R. Schettini. Noisy images-JPEG compressed: subjective and objective image quality evaluation. In S. Triantaphillidou and M.-C. Larabi, editors, *Image Quality and System Performance XI*, volume 9016, pages 274 – 282. International Society for Optics and Photonics, SPIE, 2014. URL https://doi.org/10.1117/12.2039273.

[46] A. Cortiana, V. Conotter, G. Boato, and F. G. B. D. Natale. Performance comparison of denoising filters for source camera identification. In *Media Watermarking, Security, and Forensics III*, volume 7880, pages 60 – 65. International Society for Optics and Photonics, SPIE, 2011. doi: 10.1117/12.872489. URL https://doi.org/10.1117/12.872489.

[47] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano, and L. Verdoliva. On the detection of synthetic images generated by diffusion models, 2022.

[48] D. Cozzolino and L. Verdoliva. Noiseprint: A cnn-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15:144–159, 2020. doi: 10.1109/TIFS.2019.2916364.

[49] D. Cozzolino, G. Poggi, and L. Verdoliva. Splicebuster: A new blind image splicing detector. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2015. doi: 10.1109/WIFS.2015.7368565.

[50] D. Cozzolino, J. Thies, A. Rossler, M. Niessner, and L. Verdoliva. Spoc: Spoofing camera fingerprints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 990–1000, June 2021.

[51] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007. doi: 10.1109/TIP.2007.901238.

[52] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 219–224, 2015.

[53] T. J. de Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *IEEE Transactions on Information Forensics and Security*, 8(7):1182–1194, 2013. doi: 10.1109/TIFS.2013.2265677.

[54] M. Delbracio, D. Kelly, M. S. Brown, and P. Milanfar. Mobile computational photography: A tour, 2021.

[55] Z. Deng, A. Gijsenij, and J. Zhang. Source camera identification using auto-white balance approximation. In *2011 International Conference on Computer Vision*, pages 57–64, 2011. doi: 10.1109/ICCV.2011.6126225.

[56] A. Desolneux, L. Moisan, and J.-M. Morel. *Gestalt Theory and Computer Vision*, pages 71–101. Springer Netherlands, Dordrecht, 2004. ISBN 978-1-4020-2081-0. doi: 10.1007/1-4020-2081-3_4. URL https://doi.org/10.1007/1-4020-2081-3_4.

[57] T.-T. Do, E. Kijak, T. Furon, and L. Amsaleg. Deluding image recognition in sift-based cbir systems. In *Proceedings of the 2nd ACM Workshop on Multimedia in forensics, Security and Intelligence*, pages 7–12, 2010.

[58] J. Dong, W. Wang, and T. Tan. Casia image tampering detection evaluation database. In *2013 IEEE China Summit and International Conference on Signal and Information Processing*, pages 422–426, 2013. doi: 10.1109/ChinaSIP.2013.6625374.

[59] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *JASA*, 90(432):1200–1224, 1995. doi: 10.1080/01621459.1995.10476626. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476626.

[60] T. Ehret and G. Facciolo. A Study of Two CNN Demosaicking Algorithms. *Image Processing On Line*, 9:220–230, 2019. doi: 10.5201/ipol.2019.274.

[61] T. Ehret, A. Davy, M. Delbracio, and J.-M. Morel. How to Reduce Anomaly Detection in Images to Anomaly Detection in Noise. *Image Processing On Line*, 9:391–412, 2019. doi: https://doi.org/10.5201/ipol.2019.263.

[62] W. Fan, K. Wang, F. Cayre, and Z. Xiong. Jpeg anti-forensics using non-parametric dct quantization noise estimation and natural image statistics. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pages 117–122, 2013.

[63] H. Farid. Digital doctoring: How to tell the real from the fake. *Significance*, 3:162 – 166, 11 2006. doi: 10.1111/j.1740-9713.2006.00197.x.

[64] H. Farid. *Photo Forensics*. The MIT Press, 2016.

[65] G. Fechner. Elemente der psychophysik, breitkopf und härtel. *Leipzig: Breitkopf und Härtel*, 1860.

[66] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23 (2):298–305, 1973. URL http://eudml.org/doc/12723.

[67] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE transactions on image processing*, 17:1737–54, 11 2008. doi: 10.1109/TIP.2008.2001399.

[68] M. Fontani and M. Barni. Hiding traces of median filtering in digital images. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1239–1243. IEEE, 2012.

[69] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2009.11.002. URL https://www.sciencedirect.com/science/article/pii/S0370157309002841.

[70] M. Gardella and P. Musé. Image Forgery Detection via Forensic Similarity Graphs. *Image Processing On Line*, 12:490–500, 2022. https://doi.org/10.5201/ipol.2022.432.

[71] M. Gardella and P. Musé. Forensic Similarity for Source Camera Model Comparison. *Image Processing On Line*, 12:480–489, 2022. https://doi.org/10.5201/ipol.2022.424.

[72] M. Gardella, P. Musé, J.-M. Morel, and M. Colom. Noisesniffer: a fully automatic image forgery detector based on noise analysis. In *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6, 2021. doi: 10.1109/IWBF50991.2021.9465095.

[73] M. Gardella, P. Musé, J.-M. Morel, and M. Colom. Forgery detection in digital images by multi-scale noise estimation. *Journal of Imaging*, 7(7), 2021. ISSN 2313-433X. doi: 10.3390/jimaging7070119. URL https://www.mdpi.com/2313-433X/7/7/119.

[74] M. Gardella, T. Nikoukhah, Y. Li, and Q. Bammey. The impact of jpeg compression on prior image noise. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2689–2693, 2022. doi: 10.1109/ICASSP43922.2022.9746060.

[75] M. Gardella, P. Musé, M. Colom, J.-M. Morel, and D. Perraud. Prnu-based source camera statistical certification. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2023. doi: 10.1109/WIFS58808.2023.10375045.

[76] P. Getreuer. Zhang-Wu Directional LMMSE Image Demosaicking. *Image Processing On Line*, 1:117–126, 2011. doi: 10.5201/ipol.2011.g_zwld.

[77] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Trans. Graph.*, 35(6):191:1–191:12, Nov. 2016. ISSN 0730-0301. doi: 10.1145/2980179.2982399. URL http://doi.acm.org/10.1145/2980179.2982399.

[78] A. Ghosh, Z. Zhong, T. E Boult, and M. Singh. Spliceradar: A learned method for blind image forensics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[79] F. Gisolf, A. Malgoezar, T. Baar, and Z. Geradts. Improving source camera identification using a simplified total variation based noise removal algorithm. *Digital Investigation*, 10:207–214, 10 2013. doi: 10.1016/j.diin.2013.08.002.

[80] O. Giudice, A. Paratore, M. Moltisanti, and S. Battiato. A classification engine for image ballistics of social data. In *Image Analysis and Processing - ICIAP 2017*, pages 625–636. Springer International Publishing, 2017. doi: 10.1007/978-3-319-68548-9_57. URL https://doi.org/10.1007%2F978-3-319-68548-9_57.

[81] T. Gloe and R. Böhme. The 'dresden image database' for benchmarking digital image forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, page 1584–1590, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605586397. doi: 10.1145/1774088.1774427. URL https://doi.org/10.1145/1774088.1774427.

[82] T. Gloe, E. Franz, and A. Winkler. Forensics for flatbed scanners. *Proceedings of SPIE - The International Society for Optical Engineering*, 6505, 02 2007. doi: 10.1117/12.704165.

[83] M. Goljan. Digital camera identification from images - estimating false acceptance probability. *Proc. 8th Int. Workshop Digital Watermarking*, pages 454–468, 01 2008.

[84] M. Goljan, J. Fridrich, and T. Filler. Large scale test of sensor fingerprint camera identification. In E. J. D. III, J. Dittmann, N. D. Memon, and P. W. Wong, editors, *Media Forensics and Security*, volume 7254, page 72540I. International Society for Optics and Photonics, SPIE, 2009. doi: 10.1117/12.805701. URL https://doi.org/10.1117/12.805701.

[85] M. Goljan, M. Chen, P. Comesaña, and J. Fridrich. Effect of compression on sensor-fingerprint based camera identification. *Electronic Imaging*, 2016:1–10, 02 2016. doi: 10.2352/ISSN.2470-1173.2016.8.MWSF-086.

[86] D. Gragnaniello, F. Marra, G. Poggi, and L. Verdoliva. Analysis of adversarial attacks against cnn-based image forgery detectors. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 967–971. IEEE, 2018.

[87] R. Grompone von Gioi and J. Jakubowicz. On computational gestalt detection thresholds. *Journal of Physiology-Paris*, 103(1):4–17, 2009. ISSN 0928-4257. doi: https://doi.org/10.1016/j.jphysparis.2009.05.002. URL https://www.sciencedirect.com/science/article/pii/S0928425709000229. Neuromathematics of Vision.

[88] R. Grompone von Gioi, C. Hessel, T. Dagobert, J.-M. Morel, and C. de Franchis. Ground Visibility in Satellite Optical Time Series Based on A Contrario Local Image Matching. *Image Processing On Line*, 11:212–233, 2021. https://doi.org/10.5201/ipol.2021.342.

[89] H. Guan, M. Kozak, E. Robertson, Y. Lee, A. N. Yates, A. Delgado, D. Zhou, T. Kheyrkhah, J. Smith, and J. Fiscus. Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 63–72, 2019. doi: 10.1109/WACVW.2019.00018.

[90] F. Guillaro, D. Cozzolino, A. Sud, N. Dufour, and L. Verdoliva. Trufor: Leveraging all-round clues for trustworthy image forgery detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20606–20615, June 2023.

[91] A. Hadmi, W. Puech, B. A. E. Said, and A. A. Ouahman. Perceptual image hashing. In M. D. Gupta, editor, *Watermarking*, chapter 2. IntechOpen, Rijeka, 2012. doi: 10.5772/37435. URL https://doi.org/10.5772/37435.

[92] A. Hadmi, W. Puech, B. Ait Es Said, and A. Ait Ouahman. A robust and secure perceptual hashing system based on a quantization step analysis. *Signal Processing: Image Communication*, 28(8):929–948, 2013. ISSN 0923-5965. doi: https://doi.org/10.1016/j.image.2012.11.009. URL https://www.sciencedirect.com/science/article/pii/S0923596512002135. SPECIAL ISSUE ON BIOLOGICALLY INSPIRED APPROACHES FOR VISUAL INFORMATION PROCESSING AND ANALYSIS.

[93] B. Hadwiger and C. Riess. The forchheim image database for camera identification in the wild. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part VI*, page 500–515, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-68779-3.

[94] J. F. Hamilton Jr and J. E. Adams Jr. Adaptive color plan interpolation in single sensor color electronic camera, May 13 1997. US Patent 5,629,734.

[95] B. K. T. Ho, V. Y. Tseng, M. Ma, and D. T. Chen. Mathematical model to quantify JPEG block artifacts. In Y. Kim, editor, *Medical Imaging 1993: Image Capture, Formatting, and Display*, volume 1897, pages 269 – 275. International Society for Optics and Photonics, SPIE, 1993. URL https://doi.org/10.1117/12.146974.

[96] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[97] Y.-F. Hsu and S.-F. Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In *International Conference on Multimedia and Expo*, 2006.

[98] Y.-F. Hsu and S.-F. Chang. Image splicing detection using camera response function consistency and automatic segmentation. In *International Conference on Multimedia and Expo*, 2007.

[99] M. Huh, A. Liu, A. Owens, and A. A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018.

[100] C. Iakovidou, M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris. Content-aware detection of jpeg grid inconsistencies for intuitive image forensics. *Journal of Visual Communication and Image Representation*, 54:155–170, 2018.

[101] ImageMagick. Imagemagick studio llc, 2023. URL `https://imagemagick.org`.

[102] M. Iuliani, M. Fontani, and A. Piva. A leak in prnu based source identification—questioning fingerprint uniqueness. *IEEE Access*, 9:52455–52463, 2021. doi: 10.1109/ACCESS.2021.3070478.

[103] I. Jensen and A. J. Guttmann. Statistics of lattice animals (polyominoes) and polygons. *Journal of Physics A: Mathematical and General*, 33(29):L257–L263, jul 2000. doi: 10.1088/0305-4470/33/29/102.

[104] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.

[105] Y. Jiang, H. Zeng, X. Kang, and L. Liu. The game of countering jpeg anti-forensics based on the noise level estimation. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–9, 2013. doi: 10.1109/APSIPA.2013.6694156.

[106] T. Julliand, V. Nozick, and H. Talbot. Automatic image splicing detection based on noise density analysis in raw images. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 126–134. Springer, 2016.

[107] T. Julliand, V. Nozick, and H. Talbot. Image noise and digital image forensics. In Y.-Q. Shi, H. J. Kim, F. Pérez-González, and I. Echizen, editors, *Digital-Forensics and Watermarking*, pages 3–17, Cham, 2016. Springer International Publishing. ISBN 978-3-319-31960-5.

[108] T. Julliand, V. Nozick, I. Echizen, and H. Talbot. Using the noise density down projection to expose splicing in jpeg images. 2017.

[109] K. D. Kadam, S. Ahirrao, and K. Kotecha. Multiple image splicing dataset (misd): A dataset for multiple splicing. *Data*, 6(10), 2021. ISSN 2306-5729. doi: 10.3390/data6100102. URL `https://www.mdpi.com/2306-5729/6/10/102`.

[110] X. Kang, Y. Li, Z. Qu, and J. Huang. Enhancing source camera identification performance with a camera reference phase sensor pattern noise. *IEEE Trans. on Information Forensics and Security*, 7(2):393–402, 2012. doi: 10.1109/TIFS.2011.2168214.

[111] X. Kang, J. Chen, K. Lin, and P. Anjie. A context-adaptive spn predictor for trustworthy source camera identification. *EURASIP Journal on Image and Video Processing*, 2014:19, 12 2014. doi: 10.1186/1687-5281-2014-19.

[112] A. Kashyap, R. S. Parmar, M. Agrawal, and H. Gupta. An evaluation of digital image forgery detection approaches, 2017.

[113] Y. Ke, Q. Zhang, W. Min, and S. Zhang. Detecting image forgery based on noise estimation. *International Journal of Multimedia and Ubiquitous Engineering*, 9(1):325–336, 2014.

[114] E. Kee, M. K. Johnson, and H. Farid. Digital image authentication from jpeg headers. *IEEE Transactions on Information Forensics and Security*, 6(3):1066–1075, 2011. doi: 10.1109/TIFS.2011.2128309.

[115] N. Khanna, A. K. Mikkilineni, and E. J. Delp. Scanner identification using feature-based processing and analysis. 4(1), 2009. ISSN 1556-6013. doi: 10.1109/TIFS.2008.2009604. URL `https://doi.org/10.1109/TIFS.2008.2009604`.

[116] A. Kharboutly, W. Puech, G. Subsol, and D. Hoa. Improving sensor noise analysis for ct-scanner identification. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 2411–2415, 2015. doi: 10.1109/EUSIPCO.2015.7362817.

[117] D. Kim, H.-U. Jang, S.-M. Mun, S. Choi, and H.-K. Lee. Median filtered image restoration and anti-forensics using adversarial networks. *IEEE Signal Processing Letters*, 25 (2):278–282, 2017.

[118] M. Kirchner and R. Bohme. Hiding traces of resampling in digital images. *IEEE Transactions on Information Forensics and Security*, 3(4):582–592, 2008.

[119] M. Kirchner and J. Fridrich. On detection of median filtering in digital images. In N. D. Memon, J. Dittmann, A. M. Alattar, and E. J. D. III, editors, *Media Forensics and Security II*, volume 7541, page 754110. International Society for Optics and Photonics, SPIE, 2010.

[120] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[121] P. Korus and J. Huang. Multi-scale analysis strategies in prnu-based tampering localization. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 12 2016. doi: 10.1109/TIFS.2016.2636089.

[122] P. Korus and J. Huang. Evaluation of random field models in multi-modal unsupervised tampering localization. In *Proc. of IEEE Int. Workshop on Inf. Forensics and Security*, 2016.

[123] C. Kraetzer and J. Dittmann. Considerations on the benchmarking of media forensics. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 61–65, 2015. doi: 10.1109/EUSIPCO.2015.7362345.

[124] N. Krawetz and H. F. Solutions. A picture's worth. *Hacker Factor Solutions*, 6(2):2, 2007.

[125] A. Kuzin, A. Fattakhov, I. Kibardin, V. I. Iglovikov, and R. Dautov. Camera model identification using convolutional neural networks. *2018 IEEE International Conference on Big Data (Big Data)*, pages 3107–3110, 2018.

[126] M.-J. Kwon, I.-J. Yu, S.-H. Nam, and H.-K. Lee. Cat-net: Compression artifact tracing network for detection and localization of image splicing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 375–384, 2021.

[127] M.-J. Kwon, S.-H. Nam, I.-J. Yu, H.-K. Lee, and C. Kim. Learning jpeg compression artifacts for image manipulation detection and localization. *International Journal of Computer Vision*, 130(8):1875–1895, Aug. 2022. doi: 10.1007/s11263-022-01617-5.

[128] W.-L. Lau, Z.-L. Li, and K.-K. Lam. Effects of jpeg compression on image classification. *IJRS*, 24(7):1535–1544, 2003.

[129] A. Lawgaly, F. Khelifi, and A. Bouridane. Weighted averaging-based sensor pattern noise estimation for source camera identification. In *2014 IEEE ICIP*, pages 5357–5361, 2014. doi: 10.1109/ICIP.2014.7026084.

[130] N. Le and F. Retraint. An improved algorithm for digital image authentication and forgery localization using demosaicing artifacts. *IEEE Access*, 7:125038–125053, 2019. doi: 10.1109/ACCESS.2019.2938467.

[131] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012. https://doi.org/10.5201/ipol.2012.l-bm3d.

[132] M. Lebrun, M. Colom, A. Buades, and J. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(1):475–576, 2012.

[133] M. Lee and D. Kim. Robust evaluation of diffusion-based adversarial purification. *ArXiv*, abs/2303.09051:null, 2023. doi: 10.48550/arXiv.2303.09051.

[134] B. Li, T.-T. Ng, X. Li, S. Tan, and J. Huang. Revealing the trace of high-quality jpeg compression through quantization noise analysis. *IEEE Transactions on Information Forensics and Security*, 10(3):558–573, 2015. doi: 10.1109/TIFS.2015.2389148.

[135] B. Li, T.-T. Ng, X. Li, S. Tan, and J. Huang. Statistical model of jpeg noises and its application in quantization step estimation. *IEEE Transactions on Image Processing*, 24 (5):1471–1484, 2015. doi: 10.1109/TIP.2015.2405477.

[136] C.-T. Li, A. Ho, I. Amerini, R. Caldelli, V. Cappellini, F. Picchioni, and A. Piva. *Estimate of PRNU Noise Based on Different Noise Models for Source Camera Identification*, pages 9–20. 01 2012. ISBN 9781466617599. doi: 10.4018/978-1-4666-1758-2.ch002.

[137] H. Li and J. Huang. Localization of deep inpainting using high-pass fully convolutional network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[138] W. Li, Y. Yuan, and N. Yu. Passive detection of doctored jpeg image via block artifact grid extraction. *Signal Processing*, 89(9):1821–1829, 2009.

[139] LibRaw. Libraw library. URL https://www.libraw.org.

[140] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.

[141] X. Lin and C.-T. Li. Preprocessing reference sensor pattern noise via spectrum equalization. *IEEE Transactions on Information Forensics and Security*, 11(1):126–140, 2016. doi: 10.1109/TIFS.2015.2478748.

[142] Z. Lin, R. Wang, X. Tang, and H.-Y. Shum. Detecting doctored images using camera response normality and consistency. Association for Computing Machinery, Inc., March 2005. URL https://www.microsoft.com/en-us/research/publication/detecting-doctored-images-using-camera-response-normality-and-consistency/.

[143] Z. Lin, J. He, X. Tang, and C.-K. Tang. Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis. *Pattern Recognition*, 42(11):2492–2501, 2009.

[144] B. Liu and C.-M. Pun. Splicing forgery exposure in digital image by detecting noise discrepancies. *International Journal of Computer and Communication Engineering*, 4 (1):33, 2015.

[145] B.-B. Liu, Y. Hu, and H.-K. Lee. Source camera identification from significant noise residual regions. In *2010 IEEE International Conference on Image Processing*, pages 1749–1752, 2010. doi: 10.1109/ICIP.2010.5652426.

[146] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang. Noise estimation from a single image. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 901–908, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.207. URL http://dx.doi.org/10.1109/CVPR.2006.207.

[147] C. Liu, R. Szeliski, S. B. Kang, C. Zitnick, and W. Freeman. Automatic estimation and removal of noise from a single image. *IEEE transactions on pattern analysis and machine intelligence*, 30:299–314, 03 2008. doi: 10.1109/TPAMI.2007.1176.

[148] Y. Liu, Q. Guan, and X. Zhao. Copy-move forgery detection based on convolutional kernel network. *Multimedia Tools and Applications*, 77, 07 2018. doi: 10.1007/s11042-017-5374-6.

[149] O. Losson and E. Dinet. From the Sensor to Color Images. In C. Fernandez-Maloigne, F. Robert-Inacio, and L. Macaire, editors, *Digital Color - Acquisition, Perception, Coding and Rendering*, Digital Image and Signal Processing series, pages 149–185. Wiley, Mar. 2012. URL https://hal.archives-ouvertes.fr/hal-00705825.

[150] F. Lugstein, S. Baier, G. Bachinger, and A. Uhl. Prnu-based deepfake detection. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '21, page 7–12, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382953. doi: 10.1145/3437880.3460400. URL https://doi.org/10.1145/3437880.3460400.

[151] J. Lukás, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1:205 – 214, 07 2006. doi: 10.1109/TIFS.2006.873602.

[152] S. Luo and W. Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, June 2021.

[153] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, dec 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z. URL https://doi.org/10.1007/s11222-007-9033-z.

[154] S. Lyu, X. Pan, and X. Zhang. Exposing region splicing forgeries with blind local noise estimation. *Int. J. Comput. Vision*, 110(2):202–221, Nov. 2014. ISSN 0920-5691. doi: 10.1007/s11263-013-0688-y. URL https://doi.org/10.1007/s11263-013-0688-y.

[155] B. Mahdian and S. Saic. Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 27:1497–1503, 09 2009. doi: 10.1016/j.imavis.2009.02.001.

[156] G. Mahfoudi, B. Tajini, F. Retraint, F. Morain-Nicolier, J.-L. Dugelay, and M. Pic. Defacto: Image and face manipulation dataset. In *27th European Signal Processing Conference (EUSIPCO 2019)*, A Coruña, Spain, Sept. 2019.

[157] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro. Training cnns in presence of jpeg compression: Multimedia forensics vs computer vision. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2020.

[158] H. Mareen, D. V. Bussche, F. Guillaro, D. Cozzolino, G. Van Wallendael, P. Lambert, and L. Verdoliva. Comprint: Image forgery detection and localization using compression fingerprints. *arXiv preprint arXiv:2210.02227*, 2022.

[159] F. Marra, G. Poggi, F. Roli, C. Sansone, and L. Verdoliva. Counter-forensics in machine learning based forgery detection. In *Media Watermarking, Security, and Forensics 2015*, volume 9409, page 94090L. International Society for Optics and Photonics, 2015.

[160] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi. A full-image full-resolution end-to-end-trainable cnn framework for image forgery detection, 2019.

[161] B. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975. ISSN 0005-2795. doi: https://doi.org/10.1016/0005-2795(75)90109-9. URL `https://www.sciencedirect.com/science/article/pii/0005279575901099`.

[162] O. Mayer and M. C. Stamm. Forensic similarity for digital images. *IEEE Transactions on Information Forensics and Security*, 2019. ISSN 1556-6013. doi: 10.1109/TIFS.2019.2924552.

[163] O. Mayer and M. C. Stamm. Exposing fake images with forensic similarity graphs. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):1049–1064, 2020. doi: 10.1109/JSTSP.2020.3001516.

[164] O. Mayer, B. Bayar, and M. C. Stamm. Learning unified deep-features for multiple forensic tasks. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pages 79–84. ACM, 2018.

[165] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[166] M. Mihçak, I. Kozintsev, and K. Ramchandran. Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising. *1999 IEEE ICASSP*, 6: 3253–3256 vol.6, 1999.

[167] L. Min, F. Peng, and Y. Zhu. Identifying natural images and computer generated graphics based on binary similarity measures of prnu. *Multimedia Tools and Applications*, 78, 01 2019. doi: 10.1007/s11042-017-5101-3.

[168] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012. doi: 10.1109/TIP.2012.2214050.

[169] A. Mittal, R. Soundararajan, and A. C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013. doi: 10.1109/LSP.2012.2227726.

[170] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini. Detection of malevolent changes in digital video for forensic applications. In E. J. D. III and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 300 – 311. International Society for Optics and Photonics, SPIE, 2007. doi: 10.1117/12.704924. URL `https://doi.org/10.1117/12.704924`.

[171] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69:026113, 03 2004. doi: 10.1103/PhysRevE.69.026113.

[172] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006. doi: 10.1103/PhysRevE.74.036104. URL `https://link.aps.org/doi/10.1103/PhysRevE.74.036104`.

[173] T.-T. Ng and S.-F. Chang. A data set of authentic and spliced image blocks. Technical report, Columbia University, June 2004.

[174] A. Nicolaï, Q. Bammey, M. Gardella, T. Nikoukhah, O. Boulant, I. Bargiotas, N. Monzón, C. Truong, B. Kerautret, P. Monasse, and M. Colom. The approach to reproducible research of the Image Processing On Line (IPOL) journal. *Informatio*, 27(1), June 2022. doi: 10.35643/Info.27.1.7. URL `https://hal.science/hal-04122026`.

[175] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning (ICML)*, 2022.

[176] T. Nikoukhah, J. Anger, T. Ehret, M. Colom, J.-M. Morel, and R. Grompone von Gioi. Jpeg grid detection based on the number of dct zeros and its application to automatic and localized forgery detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 110–118, 2019.

[177] X. Pan, X. Zhang, and S. Lyu. Exposing image forgery with blind noise estimation. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM&Sec '11, page 15–20, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308069. doi: 10.1145/2037252.2037256. URL `https://doi.org/10.1145/2037252.2037256`.

[178] N. N. Ponomarenko, V. V. Lukin, S. K. Abramov, K. O. Egiazarian, and J. T. Astola. Blind evaluation of additive noise variance in textured images by nonlinear processing of block DCT coefficients. In E. R. Dougherty, J. T. Astola, and K. O. Egiazarian, editors, *Image Processing: Algorithms and Systems II*, volume 5014, pages 178 – 189. International Society for Optics and Photonics, SPIE, 2003. doi: https://doi.org/10.1117/12.477717.

[179] N. N. Ponomarenko, V. V. Lukin, M. Zriakhov, A. Kaarna, and J. Astola. An automatic approach to lossy compression of aviris images. *2007 IEEE International Geoscience and Remote Sensing Symposium*, pages 472–475, 2007.

[180] A. C. Popescu and H. Farid. Statistical tools for digital forensics. In *Information Hiding*, 2004.

[181] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Signal Processing*, 53(10):3948–3959, 2005. doi: 10.1109/TSP.2005.855406.

[182] V. Prakash, K. Prasad, and T. Prasad. Color image demosaicing using sparse based radial basis function network, 09 2016.

[183] W. Puech. *Sécurité multimédia 1*. Number v. 1 in Encyclopédie sciences, Image, Compression, codage et protection des images et vidéos. ISTE editions, 2021. ISBN 9781789480269. URL https://books.google.es/books?id=2Ws3EAAAQBAJ.

[184] C.-M. Pun, B. Liu, and X. Yuan. Multi-scale noise estimation for image splicing forgery detection. *Journal of Visual Communication and Image Representation*, 38, 03 2016. doi: 10.1016/j.jvcir.2016.03.005.

[185] S. Pyatykh, J. Hesser, and L. Zheng. Image noise level estimation by principal component analysis. *IEEE transactions on image processing*, 22(2):687–699, 2012.

[186] T. Qiao, F. Retraint, R. Cogranne, and T. H. Thai. Individual camera device identification from jpeg images. *Signal Processing: Image Communication*, 52:74–86, 2017. ISSN 0923-5965. doi: https://doi.org/10.1016/j.image.2016.12.011. URL https://www.sciencedirect.com/science/article/pii/S0923596516301953.

[187] M. A. Qureshi and M. Deriche. A review on copy move image forgery detection techniques. In *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, pages 1–5. IEEE, 2014.

[188] Y. Rao, J. Ni, and H. Zhao. Deep learning local descriptor for image splicing detection and localization. *IEEE Access*, 8:25611–25625, 2020.

[189] Y. Rodriguez-Ortega, D. M. Ballesteros, and D. Renza. Copy-move forgery detection (cmfd) using deep learning for image and video forensics. *Journal of Imaging*, 7(3), 2021. ISSN 2313-433X. doi: 10.3390/jimaging7030059. URL https://www.mdpi.com/2313-433X/7/3/59.

[190] R. Salloum, Y. Ren, and C. J. Kuo. Image splicing localization using A multi-task fully convolutional network (MFCN). *CoRR*, abs/1709.02016, 2017. URL http://arxiv.org/abs/1709.02016.

[191] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-.

[192] H. J. Shin, J. J. Jeon, and I. K. Eom. Color filter array pattern identification using variance of color difference image. *Journal of Electronic Imaging*, 26(4):1 – 12, 2017. doi: 10.1117/1.JEI.26.4.043015. URL https://doi.org/10.1117/1.JEI.26.4.043015.

[193] D. Shullani, M. Fontani, M. Iuliani, O. Alshaya, and A. Piva. Vision: a video and image dataset for source identification. *EURASIP Journal on Information Security*, 2017:15, 10 2017. doi: 10.1186/s13635-017-0067-2.

[194] K. Singh, A. Kansal, and G. Singh. An improved median filtering anti-forensics with better image quality and forensic undetectability. *Multidimensional Systems and Signal Processing*, 30(4):1951–1974, 2019.

[195] N. Singh, M. Jain, and S. Sharma. A survey of digital watermarking techniques. page 6, 08 2013.

[196] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[197] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[198] S. V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997. ISSN 0034-4257. doi: https://doi.org/10.1016/S0034-4257(97)00083-7. URL https://www.sciencedirect.com/science/article/pii/S0034425797000837.

[199] J. Sun, W. Nie, Z. Yu, Z. Mao, and C. Xiao. Pointdp: Diffusion-driven purification against adversarial attacks on 3d point cloud recognition. *ArXiv*, abs/2208.09801:null, 2022. doi: 10.48550/arXiv.2208.09801.

[200] T. Taburet, P. Bas, J. Fridrich, and W. Sawaya. Computing dependencies between dct coefficients for natural steganography in jpeg domain. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'19, page 57–62, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368216. doi: 10.1145/3335203.3335715. URL https://doi.org/10.1145/3335203.3335715.

[201] T. Taburet, P. Bas, W. Sawaya, and R. Cogranne. Jpeg steganography and synchronization of dct coefficients for a given development pipeline. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '20, page 139–149, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370509. doi: 10.1145/3369412.3395074. URL https://doi.org/10.1145/3369412.3395074.

[202] M. Tailanian, P. Musé, and A. Pardo. A contrario multi-scale anomaly detection method for industrial quality inspection, 2022. URL https://arxiv.org/abs/2205.11611.

[203] M. Tailanián, M. Gardella, A. Pardo, and P. Musé. Diffusion models meet image counter-forensics. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3925–3935, January 2024.

[204] D. Teyssou, J.-M. Leung, E. Apostolidis, K. Apostolidis, S. Papadopoulos, M. Zampoglou, O. Papadopoulou, and V. Mezaris. The invid plug-in: web video verification on the browser. In *Proceedings of the first international workshop on multimedia verification*, pages 23–30, 2017.

[205] T. H. Thai, R. Cogranne, and F. Retraint. Camera model identification based on hypothesis testing theory. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1747–1751, 2012.

[206] T. H. Thai, R. Cogranne, and F. Retraint. Camera model identification based on the heteroscedastic noise model. *IEEE Transactions on Image Processing*, 23(1):250–263, 2014. doi: 10.1109/TIP.2013.2290596.

[207] T. H. Thai, F. Retraint, and R. Cogranne. Camera model identification based on dct coefficient statistics. *Digit. Signal Process.*, 40(C):88–100, may 2015. ISSN 1051-2004. doi: 10.1016/j.dsp.2015.01.002. URL https://doi.org/10.1016/j.dsp.2015.01.002.

[208] C. Tian, Y. Xu, Z. Li, W. Zuo, L. Fei, and H. Liu. Attention-guided cnn for image denoising. *Neural Networks*, 124:177–129, 2020.

[209] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic. Comofod — new database for copy-move forgery detection. In *Proceedings ELMAR-2013*, pages 49–54, 2013.

[210] J. Wagner. Noise analysis for image forensics. *https://29a.ch/2015/08/21/noise-analysis-for-image-forensics.*

[211] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[212] J. Wang, Z. Lyu, D. Lin, B. Dai, and H. Fu. Guided diffusion model for adversarial purification. *ArXiv*, abs/2205.14969:null, 2022. doi: 10.48550/arXiv.2205.14969.

[213] X. Wang, S. Niu, and H. Wang. Image inpainting detection based on multi-task deep learning network. *IETE Technical Review*, 38(1):149–157, 2021. doi: 10.1080/02564602.2020.1782274. URL https://doi.org/10.1080/02564602.2020.1782274.

[214] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

[215] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen, and S. Winkler. Coverage – a novel database for copy-move forgery detection. In *IEEE International Conference on Image processing (ICIP)*, pages 161–165, 2016.

[216] B. Widrow and I. Kollár. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. 2008. ISBN 9781139472845. URL https://books.google.fr/books?id=8q-xcGeEJDwC.

[217] G. Wu, X. Kang, and K. Liu. A context adaptive predictor of sensor pattern noise for camera source identification. In *2012 IEEE ICIP*, pages 237–240, 2012. doi: 10.1109/ICIP.2012.6466839.

[218] J. Wu, Z. Wang, H. Zeng, and X. Kang. Multiple-operation image anti-forensics with wgan-gp framework. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1303–1307. IEEE, 2019.

[219] Q. Wu, H. Ye, and Y. Gu. Guided diffusion model for adversarial purification from random noise. *ArXiv*, abs/2206.10875:null, 2022. doi: 10.48550/arXiv.2206.10875.

[220] S. Wu, J. Wang, W. Ping, W. Nie, and C. Xiao. Defending against adversarial audio via diffusion model. *ArXiv*, abs/2303.01507:null, 2023. doi: 10.48550/arXiv.2303.01507.

[221] Y. Wu, W. Abd-Almageed, and P. Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9543–9552, 2019.

[222] Z.-H. Wu, M. C. Stamm, and K. R. Liu. Anti-forensics of median filtering. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3043–3047. IEEE, 2013.

[223] C. Xiao, Z. Chen, K. Jin, J. Wang, W. Nie, M. Liu, A. Anandkumar, B. Li, and D. Song. Densepure: Understanding diffusion models towards adversarial robustness. *arXiv preprint arXiv:2211.00322*, 2022.

[224] H. Yao, S. Wang, X. Zhang, C. Qin, and J. Wang. Detecting image splicing based on noise level inconsistency. *Multimedia Tools and Applications*, 76(10):12457–12479, 2017.

[225] S. Ye, Q. Sun, and E.-C. Chang. Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *2007 IEEE International Conference on Multimedia and Expo*, pages 12–15. Ieee, 2007.

[226] G. Yu and G. Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 1, 2011. doi: 10.5201/ipol.2011.ys-dct.

[227] H.-D. Yuan. Blind forensics of median filtering in digital images. *IEEE Transactions on Information Forensics and Security*, 6:1335–1345, 12 2011.

[228] Z. Yue, Q. Zhao, L. Zhang, and D. Meng. Dual adversarial network: Toward real-world noise removal and noise generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. August 2020.

[229] M. Zampoglou, S. Papadopoulos, Y. Kompatsiaris, R. Bouwmeester, and J. Spangenberg. Web and social media image forensics for news professionals. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 10.1, 2016.

[230] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris. Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications*, 76(4):4801–4834, 2017.

[231] H. Zeng and X. Kang. Fast source camera identification using content adaptive guided image filter. *Journal of Forensic Sciences*, 61(2):520–526, 2016. doi: https://doi.org/10.1111/1556-4029.13017. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/1556-4029.13017`.

[232] H. Zeng, Y. Zhan, X. Kang, and X. Lin. Image splicing localization using pca-based noise level estimation. *Multimedia Tools and Applications*, 76(4):4783–4799, 2017.

[233] H. Zeng, M. Hosseini, K. Deng, A. Peng, and M. Goljan. A comparison study of cnn denoisers on prnu extraction, 2021. preprint available at `https://arxiv.org/abs/2112.02858`.

[234] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[235] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. doi: 10.1109/TIP.2017.2662206.

[236] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Trans. on Image Processing*, 27(9):4608–4622, Sep 2018. ISSN 1941-0042. doi: 10.1109/tip.2018.2839891. URL `http://dx.doi.org/10.1109/TIP.2018.2839891`.

[237] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[238] L. Zheng, Y. Zhang, and V. Thing. A survey on image tampering and its detection in real-world photos. *Journal of Visual Communication and Image Representation*, 12 2018. doi: 10.1016/j.jvcir.2018.12.022.

[239] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1831–1839, 2017. doi: 10.1109/CVPRW.2017.229.

[240] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Learning rich features for image manipulation detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1053–1061, 2018.

[241] N. Zhu and Z. Li. Blind image splicing detection via noise level function. *Signal Processing: Image Communication*, 68:181–192, 2018. ISSN 0923-5965. doi: https://doi.org/10.1016/j.image.2018.07.012. URL `https://www.sciencedirect.com/science/article/pii/S0923596518303588`.

[242] D. Zoran and Y. Weiss. Scale invariance and noise innature image. *IEEE International Conference on Computer Vision, Kyoto, Japan.*, 2009.

Esta página ha sido intencionalmente dejada en blanco.

List of Tables

List of Figures

Esta es la última página.
Compilado el Monday 4th March, 2024.
http://iie.fing.edu.uy/