POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

FORENSICS AND COUNTER-FORENSICS METHODS FOR SOURCE DEVICE IDENTIFICATION

Doctoral Dissertation of:
Sara Mandelli

Supervisor:
Prof. Stefano Tubaro

Tutor:
Prof. Andrea Virgilio Monti Guarnieri

The Chair of the Doctoral Program:
Prof. Barbara Pernici

2019 – XXXII Cycle

Abstract

IMAGES and videos of various nature flood the web in an unbridled fashion everyday, overwhelming our social network profiles. This wildfire spreading of visual content published online can be seen as the direct consequence of a new communication paradigm, founded on immediate and effortlessly knowledge. The concept of "social" networks itself has been revolutionized during the last few years. Nowadays, they represent not only platforms for connecting people around the world but actual websites for quick information, marketing purposes and politics propaganda.

As a matter of fact, visual communication is by far the most powerful and rapid instrument to convey a message in terms of data intelligibility. Every age, group and education can access an enormous amount of data, absorb information and share new content in few seconds. This phenomenon inevitably injects potential dangers into the communication process: whenever illegal or counterfeited data are uploaded on Internet, the longer we wait, the harder is to verify the content authenticity and avoid its uncontrolled diffusion.

In this vein, performing forensics investigations on multimedia content answers the need of smart solutions for assessing data authenticity and integrity. Tampered with and neural network generated data, as well as unknown provenance and illicit material constantly fill days of forensics analysts, just to mention some examples.

In this thesis, we tackle a few of these forensics challenges, specifically focusing on source device identification problems on images and videos. As a matter of fact, determining the origin of visual data proves extremely helpful to expose copyright violations or fight distribution of illegal content (e.g., child exploitation clips and terroristic threats). Since state-of-the-art methods for image source identification may suffer from memory and temporal requirements, we explore a novel strategy leveraging convolutional neural networks (CNNs) to identify the source camera of a query image. Our approach is a valid option to preserve important data storage and computational time.

Extending image source identification strategies to videos is far from being straightforward, and several peculiar issues must be addressed. First of all, videos are typically acquired at lower resolution and stronger compression than images. Furthermore, video stabilization technologies introduce pixel misalignment in video frames which severely affects the identification performance unless suitable countermeasures are considered. In light of this, we thoroughly investigate the stabilization mechanism and propose multiple strategies for dealing with source device identification on stabilized sequences.

Finally, and just as important, the job of forensics analyst usually includes investigations taken from a counter-forensics perspective as well. In case a malicious agent manipulates visual content with the goal of hindering investigations, forensics algorithms must be ready to address the issue and be robust to the attack. In this regard, source device anonymization is the counter-forensics instance of the identification problem. We draw two possible solutions to the anonymization task, aiming at testing the robustness and spotting weaknesses of proposed identification algorithms in literature.

To face these challenges, we rely on subtle sensor traces left by each camera on its acquired content. These characteristic footprints take the form of a random noise and are unique per device, thus enabling to directly trace data back to their origins. Specifically, we exploit a unique device fingerprint known as photo-response non uniformity (PRNU), without limiting our investigations to statistical methods but opening new horizons towards automatic and data-driven approaches.

The proposed experimental campaigns are always evaluated on images and videos from well-known disclosed datasets. The achieved results draw our strategies as competitive solutions, in terms of accuracy and computational complexity accomplishments.

Contents

CHAPTER 1

Introduction

There has never been a time in human history where visual information has been as readily available as it is today. From the very beginning of mankind, visual arts have always represented a communication medium where written texts and words were not likewise effective. Naming a few examples, the goal to convey history and culture was at the heart of cave paintings, frescoes in palaces and churches, the major architectural masterpieces, and more recently of photography and filmmaking.

What makes the current visual information different from the past is not in its contents and objectives but in how quickly information spreads from one platform to another and how easy it is to produce it.

In the last 15 years, social media giants like Facebook, Instagram and Twitter have become the primary communication channel not only for worldwide connecting people, but most importantly for marketing purposes and politics propaganda. This ingredient, mixed into the diffusion of innovative techniques for readily editing digital multimedia content, brings as a consequence a wildfire data spreading on the web with easy access by any user. The immediacy of visual impact of images and videos with respect to standard written communication allows these to be available to every age, group and education. Consequently, the average time for absorbing new information and sharing it on the web is definitely reduced to few seconds. We are far from being well prepared for the social, economic and political repercussions of this event.

As an example, it has been proved the use of social media for organized manipulation campaigns in more than 40 countries in 2018 only [1]. Computational propaganda, namely "the use of algorithms, automation, and big data to shape public life" [1], is becoming mainstream for influencing global audience with social and political implications. Experts are now worried about future political elections, which are likely going to be strongly influenced by any kind of visual content circulating over the Internet, and the threat is far from over [2, 3].

Since few decades ago, pictures and footages were considered a safe and incontrovertible proof of the trustworthiness of the facts reported by these. As a matter of fact, the past has not been free from attempts of content modification, which took precisely advantage of the presumed inviolability of these kind of visual communication. Although being true for the majority of the cases, the trustworthiness assumption did not hold for a very reduced group of contents, adulterated especially for regime propaganda reasons. Such fine contents were very carefully crafted by experts' hands, and often a comparison with the original was required for noticing the potential modifications (e.g., missing, added, or altered people and objects) [4].

Nowadays, images and videos can be no more considered a reliable source of information. The experts are talking about information disorder: multimedia content running on the web is downloaded, modified and reshared so rapidly that it is getting harder everyday to identify the differences between the original content and the modified one [5]. This revolution in visual information necessarily asks for efficient and smart solutions to verify the trustworthiness of the data. Finding ways to authenticate images or videos could help minimize the impact of visual content manipulation.

Given these premises, many forensics challenges arise in identifying whether the multimedia content is original, if it has been tampered with, or, in the worst case, if it has been completely computer generated. The forgeries can range from the classical copy-move or splicing applied to images and videos [6] to more advanced techniques which are based on convolutional neural networks to generate fake content. Some examples are the popular Face Swap [7] and Face App applications which work with images, but also Deepfake [8] videos. All of these owe their birth to generative adversarial networks (GANs) [9], which are the leading technology for modifying multimedia content in a relatively easy way.

Another current issue for forensics analysts is source device identification, that is, given some multimedia content, estimating which is the actual device which captured it. Nowadays this is a very present topic, as the source of the multimedia content circulating over the Internet is most of the time unknown and uncontrolled. Determining the origin of visual data can be paramount for solving copyright infringement cases as well as pinpointing the authors of hideous crimes such as acts of terrorism or child exploitation. This is actually a very fine request, because it is required to estimate the precise source device, not just the corresponding camera model or vendor [10].

If attributing images to their correct device is a relatively known problem in forensics community, it has a few limitations when dealing with very large image databases, for example related to memory occupation and long computational times [11]. Attributing a video to its recording device is even a more challenging problem, and several peculiar issues need to be addressed to obtain satisfactory performances. First of all, unlike images, videos are almost always compressed with relatively low quality. Further issues have been rising for the last 5 years, when mid price smartphones have begun including video stabilization mechanisms, following the trend started by the most expensive ones. Such mechanisms strongly hinder the performances of standard forensics techniques for device identification since they introduce subtle misalignment between the frames [12, 13].

Therefore, if it is true that forensics investigation is a growing field of study and it is evolved from the past, there is still much to be done to move with the times. Indeed, the job of forensic analyst still remains a mouse and cat game. Once you might think the fight is over, the "adversary" finds a novel and unexpected strategy which resets the score. For instance, whether a potential attacker aims at hindering the investigations, most of the times he/she will be able to spoil the analyst work, as it is far easier to take advantages from algorithms' weaknesses than building a robust investigation method. Furthermore, even without malicious intentions, the infinite processing chain images and videos undergo after being captured makes the forensics analysis extremely complicated and often constrains the analyst to find ad-hoc strategies related to the specific case of study.

In light of this, it is of paramount relevance for the forensics analyst to tackle problems from an adversarial perspective as well. For this reason in the last few years the counter-forensics research field has been continuously gaining importance. As a matter of fact, it is a good practice to test the vulnerability of commonly used algorithms, trying to put in the attacker's shoes and forcing the algorithm to fail. Only doing so we can test the robustness of a forensic algorithm and spot its recondite weaknesses.

For instance, diametrically opposite to the source device identification issue, we find the source device anonymization task which aims at preventing the attribution of multimedia content to its original source device. In this regard, anonymizing visual data is likewise sought-after to preserve the owner privacy, e.g., in situations where content is acquired by photojournalists or human right defenders in countries at war.

Following these considerations, this thesis brings novel contributions both in forensics and counter-forensics analysis. Precisely, we develop new strategies for dealing with source device identification, tacking the problem for both images and video sequences. In order to enhance image source identification performances in terms of accuracy and computational burden, we explore data-driven strategies based on convolutional neural networks [14] and dictionary learning algorithms [15]. Being the investigation on video sequences still in its infancy, we thoroughly analyze the stabilization mechanism and propose model-based and global optimization strategies to identify the correct source device, even if the query is a compilation of multiple sequences coming from diverse sources [16, 17]. Counter-forensics investigations are proposed as well. In order to test the vulnerability of state-of-the-art identification algorithms and preserve author privacy, we propose two diverse strategies for anonymizing images such that these can be no more linked to the original source device [18, 19]. For clarity's sake, in the next sections we introduce our original contributions providing more details.

1.1 Original contributions

This thesis proposes new developments both in forensics and counter-forensics analysis, specifically focusing on the family of source device identification problems. In particular, our identification approaches fall in the so-called passive methods category. As a matter of fact, active methods require some prior information at the analyst side to identify the source device. For instance, a watermark or a signature can be injected in images and videos at acquisition time, and this may help the analyst to match the visual content with the correct device. On the contrary, passive methods do not assume any a-priori knowledge about the source. Using passive methods means to investigate only the content for extracting forensics information regarding the acquisition process and potential modifications. In our investigations, we never exploit the meta-data information but only analyze the pixel intensity values and the statistical information contained in them.

More precisely, the proposed methods make use of an intrinsic signature of the camera inevitably injected in all its acquired content [20]. This signature has the form of a random noise, and can be exploited as a unique characteristic fingerprint typical of each device. Exploiting this fingerprint we propose different technical approaches to identify the correct source of the data, ranging from more classical model-based strategies to global optimization techniques and data-driven methods.

The first part of the thesis includes all the investigated forensics methods for solving the problem of source device identification. We start facing the problem on images. As state-of-the-art solutions rely on statistical methods which may suffer from high computational times and memory occupation [11], we propose a novel and fast approach based on convolutional neural networks (CNNs) [14]. Moreover, we append a survey tackling the impact of compressions underwent by an image, being this a very present issue and actually helpful for probing the digital history of an image [15]. On videos, we extensively analyze source device identification on stabilized video sequences, and propose manifold strategies to solve the problem by varying the required complexity and level of accuracy [16]. As an application, we study source device identification on compilations of video sequences created by temporarily alignment of video frames coming from different cameras [17].

In the second part of the thesis we show methods for the counter-forensics analysis for source device identification, which in this case converts into source device anonymization. We start with a statistical method which applies regularization techniques to modify the image content in such a way to hinder the identification performances yet maintaining high the image quality [18]. Secondly, we explore a data-driven approach using CNNs to tackle the same goal in a more efficient way [19].

Eventually, the experience we maturated in dealing with forensics tasks has led us to produce additional contributions in solving similar problems even though belonging to completely distant fields of research. In this vein, the last part of the thesis regards applications on geophysical image processing, and precisely face the challenging problem of interpolation and denoising of corrupted seismic data [21, 22].

In the following lines we report a summary for each cited contribution.

1.1.1 A first CNN-based approach for source device identification

To the best of our knowledge, state-of-the-art solutions regarding source device identification on images rely on model-based approaches based on the noise traces left by the camera sensor on the acquired multimedia content [10, 20, 23]. However, statistical methods may suffer from high computational times whenever it is required to identify the source among a huge device collection, for instance when dealing with images uploaded on social networks where there is a large pool of provenance devices to be analyzed [11]. Moreover, when working on social networks, often only cropped and resized images are available. This greatly drains the performances of state-of-the-art methods, as these usually require investigations over a large pixel portion for returning acceptable identification accuracy.

Up to now, data-driven approaches based on convolutional neural networks (CNNs) have limited their studies to camera model identification [24–26]. As a matter of fact, these approaches present very accurate performances when discriminating from one model to another, but cannot identify the specific device which shot an image. In this thesis, we propose a fast CNN-based approach for source device identification on images [14]. Specifically, we investigate two scenarios: (i) given a query image and a pool of known devices, detect which device shot the image; (ii) given a query image and a single device, detect if the device shot the image. We make use of the characteristic fingerprint of each device to train a CNN able to distinguish the subtle sensor traces left on images and match them with the correct source camera. We thoroughly test our methodology selecting cameras from Vision and Dresden image databases. Our method proves to be faster than statistical methods in case a large amount of potential provenance devices is investigated. Moreover, it requires less pixel content of the query image to obtain comparable attribution accuracies.

1.1.2 Analysis of the compression problem

Being an initial step towards source device identification based on CNNs, the strategy introduced above does not consider images which underwent editing and heavy or multiple compressions. However, the largest portion of the digital content running on the web is far from pristine, being usually compressed a few times with high quality factors at least. As a consequence, exploiting our CNN-based strategy directly on this content is not straightforward, and results may be worse than expected.

Given these premises, forensic analysts might be interested in probing digital history of content published on Internet to assess its authenticity. As a matter of fact, knowing the digital history of an image could provide the analyst an additional instrument to better predict which will be the performances of its algorithms over the investigated data. For instance, a possible indicator of image integrity is the number of JPEG compressions a picture underwent. Indeed, JPEG compression is typically operated first at image inception time directly on the acquisition device. Then, it is customary re-applied every time an image is manipulated or shared through social media. For this reason, the more the applied JPEG compressions, the more the likelihood that an image underwent some editing [27].

In this thesis, we propose an algorithm to detect multiple JPEG compressions, specifically up to four coding cycles. Our approach leverages the task-driven non-negative matrix factorization model, fed with histograms of the discrete cosine transform of the image under analysis. Experimental results show the effectiveness of the method if compared with the state-of-the-art, confirming this strategy as a viable solution for detecting multiple JPEG compressions [15].

1.1.3 Source device identification on stabilized video sequences

As reported above, identifying the source camera of a video sequence is one of the first steps to be done for analyzing its digital history. Indeed, detecting which device has been used to record a video enables to trace down the owner of this digital content, thus proves extremely helpful to solve copyright infringement cases as well as to fight distribution of illicit material (e.g., child exploitation clips, terroristic threats, etc.). Currently, the most promising methods to tackle this task exploit unique noise traces left by camera sensors on acquired images [28]. However, given the recent advancements in motion stabilization of video content, robustness of sensor pattern noise-based techniques is strongly hindered [29]. Indeed, video stabilization introduces geometric transformations to video frames, thus making camera fingerprint estimation problematic with classical approaches [30].

In this thesis, we deal with the challenging problem of attributing stabilized videos to their recording source device [16]. Specifically, we propose: (i) two strategies to extract the characteristic fingerprint of a device, starting from either a set of images or stabilized video sequences; (ii) three diverse strategies to match a stabilized video sequence with a given fingerprint. The proposed methodology is tested on videos coming from a set of different smartphones, taken from the modern publicly available Vision Dataset. The conducted experiments also provide an interesting insight on the effect of modern smartphones video stabilization algorithms on specific video frames.

1.1.4 Blind detection and localization of video temporal splicing

Being able to solve source device identification problem on video sequences paves the way for a variety of challenges which are nowadays commonplace in forensics field. For instance, in recent years, not only professional users but also amateur ones have the possibility of easily editing videos. As a consequence, user generated video compilations obtained by splicing together in time different video shots spread over the Internet in an unbridled fashion.

In order to perform forensic analysis on this kind of videos, it can be useful to split the whole sequence into the set of originating shots. As video shots are seldom obtained with a single device, we propose to identify each video shot exploiting sensor-based traces. We consider a blind situation, where videos are composed by few-second shots coming from various unknown sources that have been temporally combined [17]. The focus is on blind detection and temporal localization of splicing points. The proposed methodology is tested on different cameras taken from the recently released Vision Dataset. The method is validated on both non-stabilized and stabilized videos, thus confirming the difficulty of working in the latter scenario.

1.1.5 Counter-forensics methods for image anonymization

Over the years, the forensic community has developed a series of very accurate camera attribution algorithms enabling to detect which device has been used to acquire an image with outstanding results. Many of these methods are based on noise traces left by the camera sensor on each acquired images, allowing to powerfully trace back a picture to the camera that shot it.

Studying the boundaries of image anonymization can enable analysts to be aware of the robustness of camera attribution methods in the presence of malicious attacks. In addition, when privacy is a concern, it would be desirable to anonymize photos, unlinking them from their specific device. Photojournalists in states at war, human right defenders and activists are a few examples in which preserving the privacy of data owner may be required.

In view of this, we propose two different strategies for attenuating the noise pattern left on images by the camera sensor, specifically: (i) a strategy that deletes some pixels from the image and reconstruct them by means of inpainting described as an inverse regularized problem [18]; (ii) a data-driven strategy exploiting a convolutional neural network (CNN) which edits images in a visually imperceptible way [19]. Both the two proposed methods aim at hindering the device attribution problem, removing the traces of the device fingerprint from the image under analysis. The first strategy does not require any priors about the fingerprint to remove, while the second one needs the device fingerprint for training the image anonymization network.

1.1.6 Applications to geophysical image processing

The experience maturated on denoising and inpainting techniques investigated over this thesis allowed to tackle similar problems in completely different fields of study. Motivated by the acquired knowledge on CNNs as well, we present as an appendix a method for interpolation and denoising of seismic data [21, 22]. Specifically, we exploit CNNs for the joint tasks of interpolation and random noise attenuation of 2D common shot gathers, which, from the point of view of image processing, can be interpreted as 2D images in floating point format.

Inspired by the great contributions achieved in image processing and computer vision, we investigate a particular architecture of convolutional neural network referred-to as *U-net*, which implements a convolutional autoencoder able to describe the complex features of clean and regularly sampled data for reconstructing the corrupted ones. In our numerical experiments we consider a plurality of data corruptions including diverse noise models and distributions of missing traces.

1.2 Outline

In order to ease the thesis readability, our contributions have been organized in four main chapters.

Chapter 2 introduces the forensics investigations on images. Section 2.1 defines the PRNU and Section 2.2 introduces the state-of-the-art for the source device identification problem on images. Section 2.3 presents a novel method to deal with the problem, leveraging convolutional neural networks to identify the source camera of a query image. Finally, Section 2.4 reports an analysis of the compression problem and proposes a methodology for detecting the number of JPEG compressions an image underwent [15].

In Chapter 3 we report forensics investigations on video sequences. Section 3.1 draws the state-of-the-art for source device identification on videos, introducing the technology of video stabilization and presenting the issues related to source camera attribution on stabilized sequences. Section 3.2

Table 1.1: *Description of the notation used in the thesis.*

Symbol	Description
$\mathbf{x}$	column vector
x_i	$[\mathbf{x}]_i$
$\mathbf{X}$	matrix
$\alpha \cdot \mathbf{x}$	scalar multiplication
$\mathbf{X} \circ \mathbf{Y}$	Hadamard product
$\mathbf{XY}$	matrix multiplication

presents multiple methods to deal with source device identification on stabilized videos, showing how to compute a reliable camera fingerprint from stabilized content and how to identify correctly the provenance device [16]. Section 3.3 addresses the problem of blind detection and localization of video temporal splicing [17].

In Chapter 4 we describe the counter-forensics investigations on images. Section 4.1 presents the state-of-the-art for source device anonymization on images. Section 4.2 reports an anonymization method based on pixel inpainting [18], while Section 4.3 addresses the same goal but faces the problem with a deep learning perspective [19].

Chapter 5 depicts our contributions in geophysical image processing. Section 5.1 shows the state-of-the-art for the problem of interpolation and denoising of seismic data. In Section 5.3 we present the proposed method for dealing with the problem exploiting convolutional neural networks [21, 22].

Chapter 6 includes final considerations and draws conclusions.

1.3 Notation

For the sake of clarity, we report here the notation used overall the thesis. We indicate matrices using bold-uppercase letters, e.g., $\mathbf{X}$ whose elements are defined as $[\mathbf{X}]_{i,j}$. Column vectors are written in bold-lowercase letters, e.g., $\mathbf{x}$ with elements x_i. Given a matrix $\mathbf{X}$, its column-wise unwrapped vector is also denoted by $\mathbf{x}$. Digital images are represented as 2D matrices, while video sequences are depicted as 3D matrices, in which the third dimension is the time. The scalar multiplication is denoted by the symbol $\cdot$, the Hadamard product (i.e., the element-wise product) is defined by $\circ$, and the generic matrix product is denoted without multiplication signs. Precisely, Table. 1.1 reports the symbols of all the operations used in the thesis.

1.4 List of included publications

This thesis includes contributions from a list of peer-reviewed papers we published in the last three years in international journals and conferences. We report the complete list of included works, following the order of topics presented in Section 1.1.

- [15]: S. Mandelli, N. Bonettini, P. Bestagini, V. Lipari, S. Tubaro "Multiple JPEG compression detection through task-driven non-negative matrix factorization", in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 2106-2110.
- [16]: S. Mandelli, P. Bestagini, L. Verdoliva, S. Tubaro, "Facing device attribution problem for stabilized video sequences", IEEE Transactions on Information Forensics and Security, vol. 15, pages 14-27, 2020.
- [17]: S. Mandelli, P. Bestagini, S. Tubaro, D. Cozzolino, L. Verdoliva, "Blind detection and localization of video temporal splicing exploiting sensor-based footprints", in 2018 26th European Signal Processing Conference (EUSIPCO), pages 1362-1366, IEEE, 2018.
- [18]: S. Mandelli, L. Bondi, S. Lameri, V. Lipari, P. Bestagini, S. Tubaro, "Inpainting-based camera anonymization", in 2017 IEEE International Conference on Image Processing (ICIP), pages 1522-1526. IEEE, 2017.
- [19]: N. Bonettini, L. Bondi, D. Güera, S. Mandelli, P. Bestagini, S. Tubaro, E.J. Delp, "Fooling PRNU-Based Detectors Through Convolutional Neural Networks", in 2018 26th European Signal Processing Conference (EUSIPCO), pages 957-961, IEEE, 2018.
- [21]: S. Mandelli, F. Borra, V. Lipari, P. Bestagini, A. Sarti, S. Tubaro, "Seismic data interpolation through convolutional autoencoder", in SEG Technical Program Expanded Abstracts 2018, 4101-4105.

CHAPTER 2

Source device identification on images

In this chapter, we include the investigations regarding forensics methods for source device identification on images. Specifically, in the first part we introduce the concept of photo-response non uniformity (PRNU) and present the standard workflow used to solve the problem. Later we describe our novelties with respect to the literature, showing how data-driven approaches can outperform model-based strategies.

In this vein, we explore convolutional neural networks to satisfy important storage and temporal requirements which are usually disregarded by statistical methods. We work in the challenging scenario of tracing back to their origin images subjected to cropping operations, investigating among a large collection of provenance devices.

As the vast majority of visual data flowing over the Internet is typically edited and compressed more than once, in the last part of this chapter we provide an analysis of the impact of compression on digital images. Furthermore, with the goal of helping the forensics analyst to correctly assess their authenticity and integrity, we tackle the problem of estimating the number of JPEG compression an image underwent. We propose an algorithm leveraging task-driven non-negative matrix factorization for identifying up to 4 coding cycles.

2.1 Photo-response non uniformity

Photo-response non uniformity (PRNU) is a noise fingerprint characteristic of any image and video acquisition device. The PRNU pattern is introduced by dishomogeneities in silicon wafers and imperfections in the sensor manufacturing process, which cause a non-uniform sensitivity to light of the sensor photo-diodes. As a result, excluding saturated acquisition setup and very strong image compression, PRNU is introduced in the acquired images and video frames as a pixel-wise multiplicative zero-mean noise pattern. The classic pipeline assumes the availability of a certain number of images coming from the same device to carry out a reliable PRNU estimation. In the basic procedure proposed in the literature [10], PRNU is estimated from a set of N images $\mathbf{I}_n$ as

$$\mathbf{K} = \frac{\sum_{n=1}^{N} \mathbf{W}_n \circ \mathbf{I}_n}{\sum_{n=1}^{N} \mathbf{I}_n^2}, \tag{2.1}$$

where ratio and raising to power are element-wise and $\mathbf{W}_n$ is the noise residual extracted from $\mathbf{I}_n$. Precisely, $\mathbf{W}_n = \mathbf{I}_n - \dot{\mathbf{I}}_n$, being $\dot{\mathbf{I}}_n$ a denoised version of $\mathbf{I}_n$ computed as suggested in [10].

It has been demonstrated that, in order to compute a reliable device fingerprint, at least 25 images depicting flat scenes (e.g., sky, ground, walls, etc.) are sufficient, whereas we would need more than 50 images depicting natural content (e.g., indoor and outdoor scenes) [23, 31] to achieve a similar accuracy. For this reason, if available, flat scenes are always recommended for PRNU estimation. Since PRNU is a multiplicative noise, totally dark or saturated pixels bring no information about it. The same rationale occurs for images undergone to strong JPEG compressions, which may hinder the possibility of finding some residual content in pixel values.

Due to its properties, the PRNU pattern allows reliable device identification [20]. Moreover, it can be used for other forensic tasks, such as image forgery detection [10, 32]. PRNU-based approaches normally rely on some prior information, typically a large number of images known to come from the camera of interest. However, blind methods have also been proposed with competitive performances [33, 34]. In addition, the use of compressed PRNU patterns has been proposed in [11, 35] to allow real-time applications.

In the next section, we provide more details on the use of PRNU for source device identification.

2.2 State-of-the-art for image-camera attribution problem

State-of-the-art solutions use PRNU $\mathbf{K}$ as a unique camera fingerprint to solve image-camera attribution problem. Given an image **I** and a test device d, it is possible to solve the problem using statistical methods exploiting the PRNU and the noise residual extracted from the image.

In particular, one way to solve the problem is based on measuring the normalized cross-correlation between the image noise residual $\mathbf{W}$ and the device PRNU $\mathbf{K}_d$ pixel-wise scaled by **I**. The normalized cross-correlation (NCC) between these terms can be expressed as:

$$\mathrm{NCC}(\mathbf{W}, \mathbf{K}_d \circ \mathbf{I}) = \frac{(\mathbf{w} - \mu_{[\mathbf{w}]})^{\mathrm{T}}(\mathbf{k}_d \circ \mathbf{i} - \mu_{[\mathbf{k}_d \circ \mathbf{i}]})}{||\mathbf{w} - \mu_{[\mathbf{w}]}||_2 \cdot ||\mathbf{k}_d \circ \mathbf{i} - \mu_{[\mathbf{k}_d \circ \mathbf{i}]}||_2}, \tag{2.2}$$

being $|| \cdot ||_2$ and $\mu_{[\cdot]}$ the symbols for ℓ_2 norm and mean value, respectively.

A more general solution is given by the peak to correlation energy (PCE) [23]. In order to compute it over matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ of size $m \times n$, we first have to cross-correlate them, defining $[\mathbf{R}]_{u,v} = \mathrm{corr}([\mathbf{W}_1]_{i,j}, [\mathbf{W}_2]_{i-u,j-v})$. The $\mathrm{PCE}(\mathbf{W}_1, \mathbf{W}_2)$ is then defined as

$$\mathrm{PCE}(\mathbf{W}_1, \mathbf{W}_2) = \frac{[\mathbf{R}]^2_{u,v}}{\frac{1}{m \cdot n - |\mathcal{N}_p|} \cdot \sum\limits_{u,v \notin \mathcal{N}_p} [\mathbf{R}]^2_{u,v}}, \tag{2.3}$$

where (u_p, v_p) are the coordinates of the maximum correlation peak, $\mathcal{N}_p$ is a small neighborhood of the peak and $|\mathcal{N}_p|$ its cardinality [23]. In particular, (u_p, v_p) can be seen as an estimate of the mutual shift between $\mathbf{W}_1$ and $\mathbf{W}_2$. Dealing with the standard image-camera attribution problem, we can compute the PCE between the noise residual $\mathbf{W}$ and the camera PRNU pixel-wise scaled by **I**, denoted as $\mathrm{PCE}(\mathbf{W}, \mathbf{K}_d \circ \mathbf{I})$.

Considering both the two metrics, we predict that **I** has been taken by the device d if the obtained NCC or PCE are higher than a confidence threshold, set in order to bound false-detection probability below a chosen value [10,20]. Actually, the dynamics of these metrics is very far one to the other, thus the threshold modifies its values according to the used metrics. For instance, NCC is always limited to 1, whereas PCE can extend up to 10^3 or more.

The presented metrics can be used almost alternatively according to the specific tackled issue. If the analyzed images have the same resolution of the camera PRNUs and did not undergo geometrical transformations (e.g., translations, rotations, etc.), the NCC may be the favourite metrics as it requires only one scalar product per image, therefore it is computationally cheap. On the contrary, PCE is very robust to potential shifts between images and PRNUs. This property helps attributing images that have been cropped with respect to the reference PRNU [36]. Furthermore, PCE has been proven to be slightly more accurate than NCC even when images and PRNU did not undergo any misalignment [23]. As a matter of fact, (2.3) evaluates the energy of the maximum peak with respect to the average energy of the remaining cross-correlation values, instead of limiting the evaluation to the peak only. In doing so, PCE helps avoiding undesirable errors which may occur whenever cross-correlation matrix is noisy and does not show a unique pronounced peak. However, evaluating PCE comes at the expense of increased computational cost, since PCE needs to compute the complete cross-correlation matrix $\mathbf{R}$ (considering every possible shift) between $\mathbf{W}$ and $\mathbf{K}_d \circ \mathbf{I}$.

2.3 Image source device identification with CNNs

As reported in Section 2.1, the classic pipeline for source device identification on images assumes the availability of a certain number of images coming from the same device to carry out a reliable PRNU estimation. Then the noise residual of the image under test is computed and compared with the sensor fingerprint usually by measuring the normalized cross-correlation (NCC) or the peak-to-correlation energy (PCE).

However, solutions based on statistical methods may suffer from high computational times whenever it is required to identify the source among a large device collection, as it is typical for the goals of source identification or image-to-identity linking over social networks [34,37,38]. Furthermore, even though the size of analyzed image portion contributes in enhancing the attribution accuracy, it increases the identification time as well: the larger the tested pixel-area, the higher the required identification time.

Beyond these considerations, it is important to highlight that when working on social networks imagery, often only cropped and resized images are available [39] [40].

Recent works on convolutional neural networks (CNNs) propose to solve the camera-model identification problem [25,26,41]. Authors exploit CNNs to extract model-related features which enable an accurate model identification. Nonetheless, so far CNN-based methods have shown low accuracy in identifying the very precise device in a set of devices taken from the same camera model.

Given these premises, we propose a first solution to deal with the image source device identification problem exploiting CNNs. Specifically, we in-

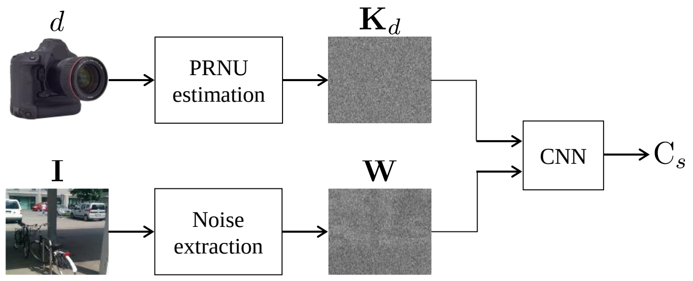


Figure 2.1: *Sketch of the proposed method for source device identification based on CNNs.*

vestigate two situations:

1. *closed-set* scenario: given an image and a set of known devices, inferring which device shot the image;

2. *open-set* scenario: given an image and one device, inferring if the device shot the image.

To solve these problems, we do not resort to the cross-correlation tests shown in 2.2, we exploit a particular CNN architecture trained with the specific purpose of source device identification instead. We test our method selecting devices from Dresden image database [42] and from the recently released Vision dataset [29], showing significant performance enhancement with respect to statistical methods, especially when a large amount of potential source devices is analyzed. Focusing on a reduced pixel region, the proposed methodology is able to save storage, at the same time reaching comparable accuracy to state-of-the-art. It follows a detailed explanation of the proposed method.

2.3.1 Proposed method

Likewise statistical methods, in order to solve the identification problem, we exploit the device PRNU as unique device fingerprint and the noise residuals extracted from images as reported in [10]. Opposite to state-of-the-art, we aim at avoiding the cross-correlation test in favour of an alternative attribution metrics based on CNNs. In a nutshell, as depicted in Figure 2.1, for each pair of query image $\mathbf{I}$ and candidate device d, we feed the PRNU $\mathbf{K}_d$ and image noise residual $\mathbf{W}$ to the CNN. The network returns a CNN-based identification score C_s which is directly associated to the coherence between image and device. By thresholding C_s we can infer whether $\mathbf{I}$ comes from device d or not.

In the following, we describe the dataset generation and the CNN architectures used in the experiments, motivating our choice and drawing details about the selected train, validation and test strategies.

2.3.2 Dataset creation

As in any supervised problem, we assume to have training and validation data used by the CNN to learn the best parameters' configuration for solving the problem. Considering a pool of N_d devices, we first compute the devices' PRNUs and the noise residual $\mathbf{W}$ for every available image. Then, we randomly extract three disjoint noise residuals' datasets from each device: the training dataset $\mathcal{W}_T$ including 50% of the data, the validation dataset $\mathcal{W}_V$ containing 25% of the residues and the evaluation dataset $\mathcal{W}_E$ built with the last 25%. Finally, we create dataset $\mathcal{D}_{TV}$ which includes a reduced amount of devices used for training and validating the network.

As depicted in Figure 2.1, the input to the network is always a pair of PRNU and image noise residual, concatenated along the so-called "channel" dimension (i.e., the third dimension). We analyze only a central image portion of $P \times P$ pixels with the goal of limiting the network complexity. Moreover, the central pixel region should be characterized by lower misalignment between PRNU and residue, thus it likely to produce better identification performances than image corners.

As commonly done in CNN-based solutions, we normalize both PRNUs and residues by their standard deviation before to feed them to the network.

2.3.3 Proposed CNN architectures

We investigate two CNN architectures rather different one from the other.

With the specific purpose of testing a shallow network enabling fast computations, the first architecture is drawn using only 3 convolutional layers as depicted in Figure 2.2. For each pair $(\mathbf{K}_d, \mathbf{W})$, we follow these operations:

1. feed the pair to 3 steps of 2D Convolution, Leaky Relu and Max Pooling, using the parameters shown in Figure 2.2;
2. add a Pair-wise Correlation Pooling layer, which is a layer tailored to our specific problem;
3. perform a Fully Connected layer, obtaining a single output score.

Precisely, the Pair-wise Correlation Pooling layer is inspired to the conventional metrics used to identify the source device on images, i.e., NCC

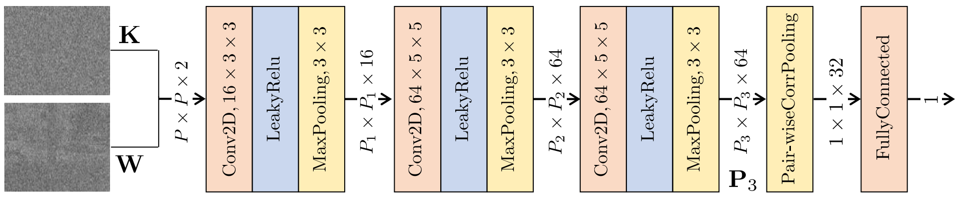


Figure 2.2: *Sketch of the proposed pair-wise correlation network (PCN).*

and PCE. Since these are based on cross-correlation between noise residual and PRNU, this layer computes a correlation as well between adjacent pairs of input features. Being $\mathbf{P}_3$ the input of Pair-wise Correlation Pooling, the output is defined as:

$$[\mathbf{P}_4]_n = \frac{1}{P_3^2} \sum_{p_i=1}^{P_3} \sum_{p_j=1}^{P_3} [\mathbf{P}_3]_{p_i,p_j,2n-1} \cdot [\mathbf{P}_3]_{p_i,p_j,2n} \quad n \in [1, 32], \tag{2.4}$$

where n is the channel index.

The Pair-wise Correlation Pooling layer can be seen as a simplified version of Bilinear Pooling layer. Actually, Bilinear Pooling computes the correlation between *all* the features maps, whereas Pair-wise Pooling evaluate this only for *adjacent* pairs of features. The main motivation behind Pair-wise Pooling is providing fast computations. Indeed, evaluating the complete cross-correlation between all feature maps may be time-consuming.

The second network we propose is known in state-of-the-art as Inception-ResNet V2 [43]. Diametrically opposite to the previously shown architecture, Inception-ResNet V2 is designed for going deeper and providing very accurate results. Differently from the first solution which is trained from scratch, we initialize the network weights, except for the last layer, using the weights trained on Imagenet database [44]. Since Inception-ResNet V2 works on RGB images, we use respectively the R and G channels for the PRNU and the residue.

For the sake of clarity, we denote the two network architectures as PCN (pair-wise correlation network) and INC (inception).

2.3.4 Network training

We train the network using devices in dataset $\mathcal{D}_{TV}$. The network is trained on coherent pairs (PRNU and residue of the same device), and non-coherent pairs (PRNU and residue of diverse devices). In order to provide both coherent and non-coherent cases for each device inside one batch of data, we consider a batch-size of twice the cardinality of $\mathcal{D}_{TV}$. For every $d \in \mathcal{D}_{TV}$, we build the pairs using two slightly different approaches:

1. the coherent pair $(\mathbf{K}_d, \mathbf{W}_d)$ is created by randomly picking $\mathbf{W}_d$ from the set of residues of device d, while the non-coherent pair $(\mathbf{K}_d, \mathbf{W}_{\bar{d}})$ randomly selects $\mathbf{W}_{\bar{d}}$ from another device $\bar{d}$ different from d, $\bar{d} \in \mathcal{D}_{TV}$;
2. the coherent pair $(\mathbf{K}_d, \mathbf{W}_d)$ is created as above, while the non-coherent pair $(\mathbf{K}_{\tilde{d}}, \mathbf{W}_d)$ takes $\mathbf{K}_{\tilde{d}}$ from another device $\tilde{d}$ different from d, $\tilde{d} \in \mathcal{D}_{TV}$.

Label 1 is assigned to coherent pairs and label 0 otherwise. The network returns a score $\mathrm{C}_{s_\mathsf{c}}$ for coherent pairs and score $\mathrm{C}_{s_\mathsf{nc}}$ for non-coherent ones.

It is worth noting that the first strategy forces the presence of PRNU $\mathbf{K}_d$ in both coherent and non-coherent pairs, whereas the second one maintains $\mathbf{K}_d$ only in the coherent case. Since in testing phase the network receives a generic pair of PRNU and residue, seeing twice the same PRNU during training is likely to be more accurate, although being a more constrained approach. For the ease of reference, the two strategies are named as pK (pivot-on-PRNU $\mathbf{K}_d$) and pW (pivot-on-residue $\mathbf{W}_d$), respectively, from the term which maintains unaltered inside coherent and non-coherent pairs.

As far as the loss function is concerned, we investigate two strategies as well. We tackle the problem as a binary classification problem aiming at identifying coherent pairs versus non-coherent ones. As first strategy we adopt the standard sigmoid cross-entropy loss, defined as:

$$l_{\mathrm{CE}} = \sum_{\mathrm{C}_{s_\mathsf{c}} \in \mathcal{C}_{s_\mathsf{c}}} \ln\left(1 + e^{-\mathrm{C}_{s_\mathsf{c}}}\right) + \sum_{\mathrm{C}_{s_\mathsf{nc}} \in \mathcal{C}_{s_\mathsf{nc}}} \ln\left(1 + e^{\mathrm{C}_{s_\mathsf{nc}}}\right) \tag{2.5}$$

where $\mathcal{C}_{s_\mathsf{c}}$ and $\mathcal{C}_{s_\mathsf{nc}}$ are respectively the set of scores related to coherent and non-coherent pairs inside one batch.

The second solution has been inspired by [45, 46] which show possible loss functions related to the area under the ROC curve of the binary classification problem. In detail, relying on the so-called softplus function [47], we define the loss as:

$$l_{\mathrm{AUC}} = \sum_{\mathrm{C}_{s_\mathsf{c}} \in \mathcal{C}_{s_\mathsf{c}}, \mathrm{C}_{s_\mathsf{nc}} \in \mathcal{C}_{s_\mathsf{nc}}} \ln\left(1 + e^{\mathrm{C}_{s_\mathsf{nc}} - \mathrm{C}_{s_\mathsf{c}}}\right). \tag{2.6}$$

As in standard neural network training, we follow an iterative procedure to minimize either (2.5) or (2.6), stopping at the iteration where the accuracy (i.e., the fraction of correct predictions) over the validation set

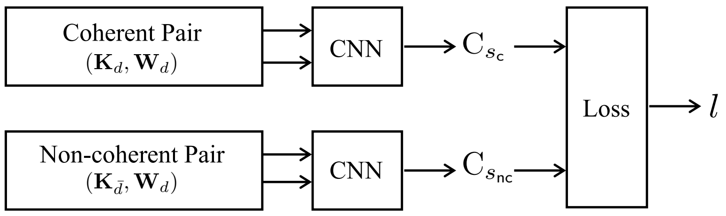

Coherent Pair
$(\mathbf{K}_d, \mathbf{W}_d)$
CNN
C_{s_c}
Loss
l
Non-coherent Pair
$(\mathbf{K}_{\bar{d}}, \mathbf{W}_d)$
CNN
$C_{s_{nc}}$

Figure 2.3: *Sketch of the proposed training strategy. For every device $d \in \mathcal{D}_{TV}$, one coherent and one non-coherent pairs of PRNU-residue feed the network.*

is maximum. Specifically, we use Adam optimization algorithm [48] with learning rate and patience initialized at 0.001 and 30, respectively, training the network for a maximum number of 500 epochs.

To clarify, Figure 2.3 depicts a scheme of the proposed training strategy.

2.3.5 Network deployment

In testing phase, we investigate two scenarios:

1. a *closed-set* scenario, in which, given an image, the forensics analyst must identify the source among a finite pool of devices;

2. an *open-set* scenario, in which, given an image and a candidate device, the forensic analyst must infer if the device shot that image or not.

To solve these problems, we always feed the network with pairs of image residue and device PRNU. In the closed-set scenario, the camera returning the highest CNN score is associated to the image. In the open-set scenario, we can threshold the CNN score in order to attribute the image to the device with a certain false alarm probability.

To explore both scenarios, we use the noise residuals belonging to evaluation dataset $\mathcal{W}_E$ for each of the N_d investigated devices. Notice that we test also cameras belonging to $\mathcal{D}_{TV}$, whose PRNUs have been already seen by the network. In doing so, we aim at verifying that the network did not learn to identify known devices only, but can generalize the attribution process to any device.

2.3.6 Experiments and results

In this section, we first describe the experimental set-up and the evaluation metrics, then we report numerical results discussing the main achievements.

Dataset

In order to test our method on a significant amount of devices, we consider both the Dresden image database [42] and the recently released Vision dataset [29]. To avoid excessively old camera models, we only pick devices whose imagery have resolution greater than 720×720 pixels. For each device we exclusively investigate JPEG compressed images, as these represent the most frequent data for a forensics analyst to deal with. Focusing on computing reliable PRNU fingerprints, we select devices if at least 25 JPEG images depicting scenes of flat surfaces are available. In doing so, we end up with 55 devices from Dresden and 32 devices from Vision.

To build the PRNU we select only flat-field images, ranging from a minimum of 25 shots per device to a maximum of 150. In order to fairly test our method, we make use of images showing natural scenes taken from indoor and outdoor scenarios. We work with more than 12000 images belonging to Dresden and almost 7000 images from Vision. In both cases, more than 200 images per device are available on average.

As reported in Section 2.3.2, before being fed to the network PRNUs and residues are cropped in the central portion by $P \times P$ pixels. We investigate $P = [64, 128, 256, 320, 512, 640, 720]$.

The training-validation dataset $\mathcal{D}_{TV}$ includes 20 devices from Vision and 16 devices from Dresden. It is worth mentioning that $\mathcal{D}_{TV}$ has been built using only devices of *different* models. In doing so, we avoid introducing an important constraint into the learning process, that is, training using cameras of the same model. Indeed, this may help enhancing the final performances, although requiring various instances of the same model at investigation side, which is actually unlikely to happen. Furthermore, we remove another big constraint from the training procedure, specifically we do not include all the available camera models in $\mathcal{D}_{TV}$. Thereby, we are simulating a real situation in which the system has to be tested over unknown camera models.

Evaluation metrics

We exploit two diverse evaluation metrics according to the considered scenario.

In the closed-set scenario we test the residue of the query image with N_d PRNUs. To assess the attribution accuracy, we include all possible query residues in evaluation set $\mathcal{W}_E$ for each device. The closed-set accuracy score $\mathrm{A}_{\textsf{cs}}$ is defined as the average fraction of correct predictions per device.

The open-set situation can be solved as a binary classification problem

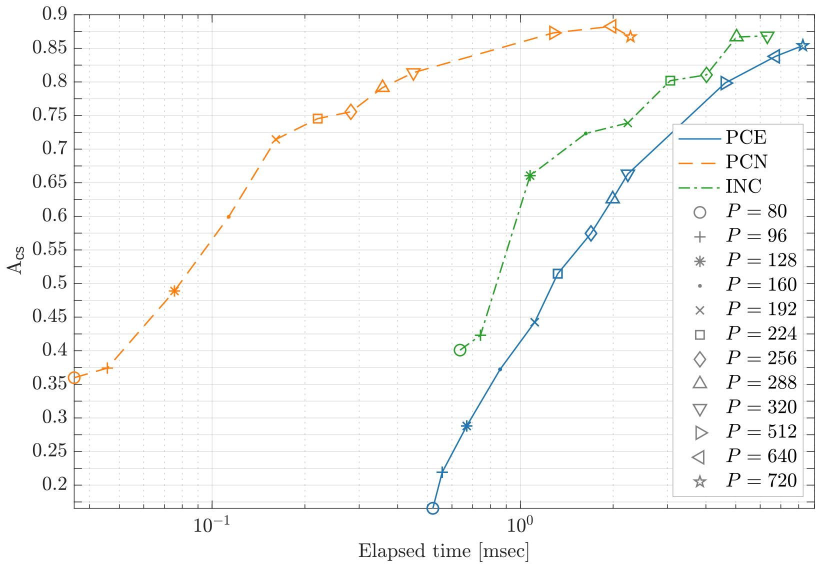


Figure 2.4: *Accuracy A_{cs} as a function of time [milliseconds] and crop size P.*

which aims at distinguish correlating pairs of residue-PRNU from non-correlating ones. To assess the accuracy, we resort to receiver operating characteristic (ROC) curves. Specifically, for each camera, we consider all noise residuals shot by that camera (i.e., the entire $\mathcal{W}_E$ set) as positive samples, whereas the set of negatives includes an equal amount of residues not taken with that camera, randomly selected from the evaluation dataset. Each ROC curve draws the relationship between true positive rate (TPR) and false positive rate (FPR), averaged over the set of available devices. To numerically evaluate the accuracy, we use $\mathrm{AUC}_{\mathrm{os}}$, defined as the area under the curve (AUC) for the open-set problem. The goal is to achieve a high value of $\mathrm{AUC}_{\mathrm{os}}$, ideally 1.

Results

For each crop size P, we evaluate results as a function of the chosen network architecture, training strategy and loss function. We performed a comprehensive investigation of all these parameter combinations, obtaining comparable results among them. For the sake of brevity, we show results only for two set-ups which tackle the problem from diverse perspectives:

1. exploit the PCN architecture with pK training strategy and l_{CE} loss;

2. exploit the INC architecture with pW training strategy and l_{AUC} loss.

The former method proposes a more customary and leaner approach: PCN is not a complex architecture thus enables fast computations, and pK and l_{CE} can be viewed as the most intuitive strategies to deal with a CNN-based classification problem involving PRNU. On the contrary, the latter method proposes a deeper architecture, an unconventional training strategy (pW) and a more sophisticated loss function. We expect this method to provide higher accuracy at the expense of increased computational complexity. As state-of-the-art comparison, we perform the PCE test between each noise residual in the evaluation dataset $\mathcal{W}_E$ and the $N_d = 87$ PRNUs of devices. Following the considerations done in Section 2.2, we choose PCE instead of NCC in order to guarantee the best attribution performances.

Figure 2.4 shows results for the closed-set scenario. In particular, A_{cs} is depicted as a function of the average computational time for testing one pair of PRNU-residue and of the crop size P. Notice that for the second parameter set-up (i.e., INC, pW, l_{AUC}) we draw results only for $128 \leq P \leq 320$, as $P = 64$ is too small to feed INC architecture and larger image dimensions (i.e., $P \geq 512$) would require additional GPU memory. For the sake of clarity, our workstation is composed by one Intel® Core™ i9-9980XE (36 Cores @3.00 GHz), RAM 126 GB, running Ubuntu 18.04.2, and one QUADRO P6000 (3840 CUDA Cores @1530 MHz), 24 GB.

As previously announced, the first parameter set-up (i.e., PCN, pK, ℓ_{CE}) reveals to be fast and accurate at the same time: indeed, for equal values of P, the required time is always at least 4 times slower than PCE and 10 times slower than second strategy. With respect to PCE, accuracy grows as well: indeed, PCE is always worst and gets similar results only for the highest P values. Second strategy is computationally heavier as expected, but shows an accuracy gap with respect to PCE that is larger than the previous method. For instance, considering $P = 256$, the second set-up achieves $\mathrm{A}_{\mathsf{cs}} = 0.86$ while first set-up only reaches 0.76 and PCE obtains 0.64. To achieve similar accuracies, PCE should consider an image size of more than twice that of second strategy.

For what concerns the required computational times, it is worth noting that the main advantage with respect to PCE is the possibility of feeding the network with multiple pairs of residue-PRNU. These pairs can be processed in parallel, at least until there is available GPU memory for storing the data. As a consequence, the larger the amount of candidate devices, the higher the CNN temporal benefit if compared to PCE. For instance, when testing a query image over $N_d = 87$ candidate cameras, we take up less than 8 GB of GPU memory using the second parameter configuration. Whether more devices were available, the average testing time would remain basically

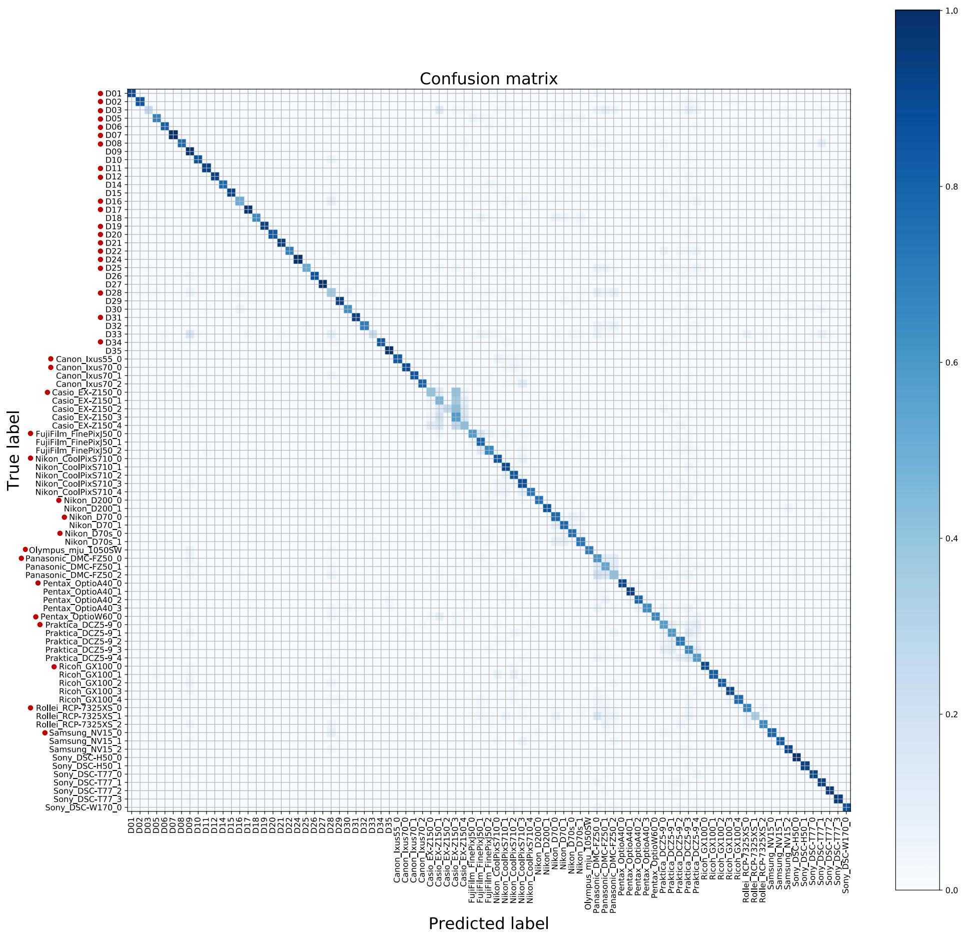


Figure 2.5: *Confusion matrix of the closed-set problem, for $P = 256$, considering PCN, pW and l_{CE} as parameter configuration. Devices in $\mathcal{D}_{TV}$ are denoted by* ●.

unchanged until reaching the maximum memory size.

In order to verify that we do not confuse instances of the same camera model, we analyze the confusion matrix for each investigated parameter configuration. Specifically, each row of confusion matrix is associated to one provenance device, while columns are related to the devices predicted by the network. We fill the matrix with the resulting CNN predictions considering all noise residuals in the evaluation dataset, then we normalize each row by its cardinality.

For brevity's sake, we show results narrowing down to the case $P = 256$ which represents a good trade-off between accuracy and required storage and computational time. Figures 2.5 and 2.6 depict the confusion matrix for

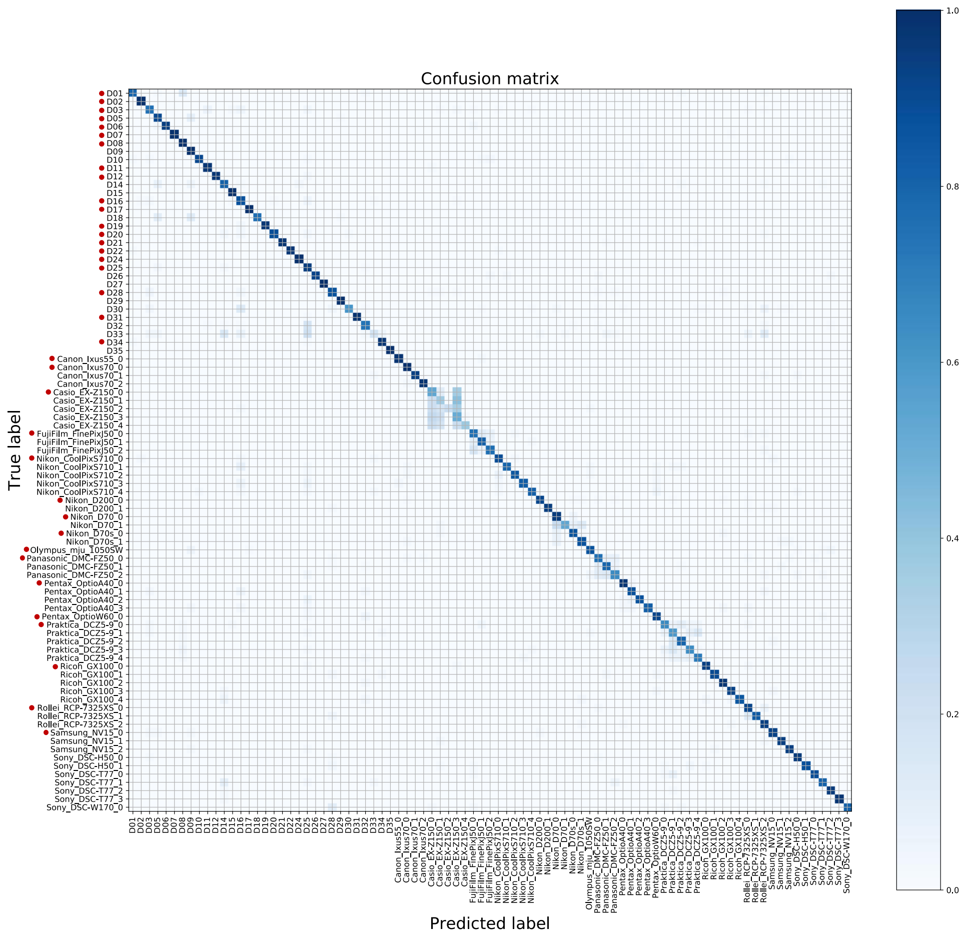


Figure 2.6: *Confusion matrix of the closed-set problem, for $P = 256$, considering INC, pK and l_{AUC} as parameter configuration. Devices in $\mathcal{D}_{TV}$ are denoted by* ●.

the first and second parameter configuration, respectively. To be precise, we use as device nomenclature for Vision devices the one reported in [29], and we do the same for devices in [42]. Red dots correspond to devices included in training-validation set $\mathcal{D}_{TV}$. It is worth noting that both the two matrices present a diagonal behaviour, meaning that instances of the same model are not confused and identification accuracy is quite good. The only exception is represented by Casio-EX-Z150 model whose instances are slightly mixed up. Figure 2.7 shows the confusion matrix associated to PCE. Notice that instances of the same model are never confused, nonetheless there are some models which are wrongly predicted most of the times. For example, instances of Panasonic-DMC-FZ50 and Praktica-DCZ5-9 are hardly ever

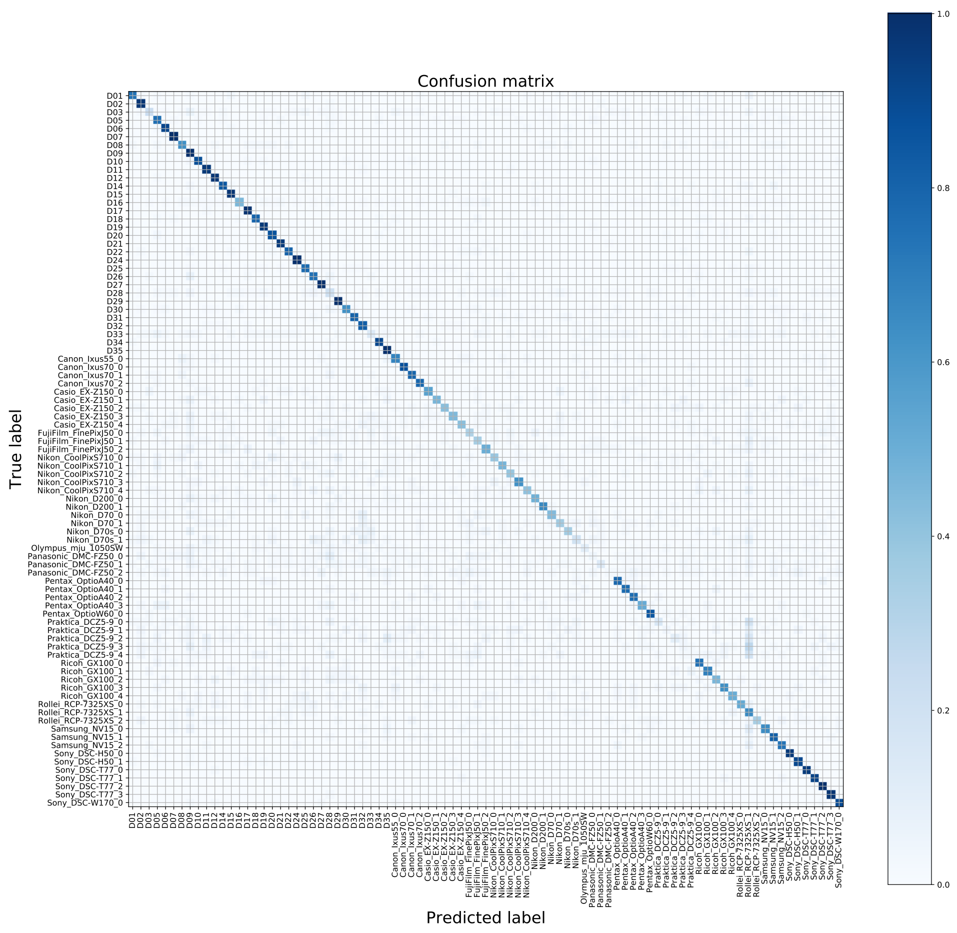


Figure 2.7: *Confusion matrix of the closed-set problem, for* $P = 256$ *and* PCE *strategy.*

identified, being mistaken for diverse models.

Results of the open-set scenario are reported in Figure 2.8. It is worth noting that both proposed methods exceed PCE for any $P \geq 128$. Only for $P = 64$ the proposed method has a performance decay but it is anyway superior to the PCE evaluated with the same crop size.

Notice that we are not analyzing $\mathrm{AUC}_{\mathrm{os}}$ as a function of the required time, since the open-set scenario reduces investigations to only one pair of PRNU-residue. In this scenario, both CNN and PCE report comparable computational time, in the order of few milliseconds. However, as previously shown, whenever various images should be tested against the same PRNU, we could exploit the data parallelization property of GPU to test multiple images together, thus saving important computation time.

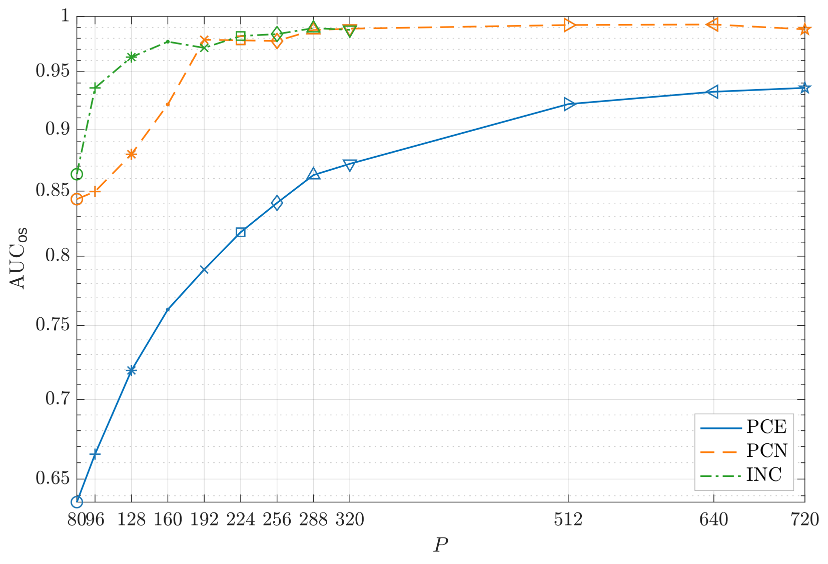


Figure 2.8: *Accuracy* $\mathrm{AUC}_{\mathrm{os}}$ *as a function of crop size* P.

2.4 Analysis of the image compression problem

As images available online are likely to be the result of a multi-processing chain, this engenders concerns about their authenticity and integrity [49]. Therefore, multimedia forensics community has been focusing on detecting potential processing traces on images with the goal of restore faith in digital pictures [4, 50]. In particular, due to the wide diffusion of JPEG image coding scheme, there is wide literature devoted to investigate and exploit traces left by JPEG compression both in forensic [51–53] and counter-forensic [54–56] scenarios. Indeed, most of the images available online are compressed according to JPEG standard, which leaves on each picture peculiar traces that can be exploited for forensic investigations.

Many of the forensic techniques proposed so far limit their goal to the detection of double compression, i.e., they aim at discovering if an image has been compressed once or twice [57–59]. Several proposed approaches are based on the investigation of quantized Discrete Cosine Transform (DCT) coefficient statistics, which are characteristically shaped by JPEG compression procedure. Some works embrace the analysis of histograms of DCT coefficients [60–62], or DCT First Significant Digit [63]. However, in many practical situations, pictures under analysis might be compressed several times. Think for example to the widespread habit of sharing visual content on social media: the average user typically shot a picture with a smartphone (first compression), share it through a messaging app (second compression), and the receiver may re-share it or post it on a social platform (third compression). As a consequence, this operation necessarily results in an "at-least-three-compression" chain for the image under analysis.

Given the strong likelihood of digital images to undergo more than two compression stages, finding a method able to estimate the number of endured JPEG compressions is of paramount importance for the reconstruction of processing history of the investigated content. In this vein, the method proposed in [64] aims at identifying up to three JPEG compressions through a testing scheme based on the statistical analysis of Benford-Fourier coefficients [65]. Authors of [66] address up to four JPEG compression detection, by exploiting the First Significant Digit of DCT coefficient in absolute values.

In light of this, we propose a novel method for detecting up to four JPEG compressions. In particular, we cast multiple JPEG compression as four-class supervised classification problem. To solve it, we exploit Task-driven Dictionary Learning (TDL) model described in [67]. The goal of TDL is to learn a feature dimensionality reduction strategy based on sparse data representation, which minimizes classification loss by jointly optimizing the dictionary used for dimensionality reduction and the classifier. More specifically, we propose to feed histograms of DCT coefficients as features to TDL model. Given the non-negativity of these features (i.e., histograms bin counts cannot be negative), we illustrate a more specific formulation of TDL, namely Task-driven Non-negative Matrix Factorization (TNMF) [68], which has proven to be even more effective than deep learning paradigms situations characterized by non-negative features [69].

2.4.1 Problem formulation

During standard JPEG compression, input images are partitioned into 8×8 non-overlapped pixel blocks. Discrete Cosine Transform (DCT) is computed for each one of them, and transform coefficients are quantized into integer-valued levels depending on the selected quantization matrix and quality factor (QF). Quantized values are then converted into a binary stream, exploiting lossless coding. In decoding phase, binary stream is decompressed, coded blocks are reconstructed by applying inverse DCT on rescaled coefficients and the image is re-built in the pixel domain [70]. Due to quantization, it is well-known in the literature that histograms of DCT coefficients show a typical comb-like shape, and spacing between consecutive peaks is related to the adopted quantization step size. Moreover, if an image is encoded many times with different quality factors, the resulting quantization levels are modified accordingly [71].

In this section we propose a method for detecting multiple JPEG compressions. Namely, given an image, detect how many times (up to four) it has been JPEG compressed. To do so, we leverage perturbations of DCT histograms that capture traces of multiple compressions, and train a supervised classifier to discriminate between images compressed different amount of times. More in detail, multiple-JPEG detection is performed using Task-driven Non-negative Matrix Factorization (TNMF) algorithm [67, 68, 72]. This method allows to reduce feature dimensionality, which helps avoiding data redundancy, by jointly estimating a dictionary for reduced data representation and a multinomial classifier for multiple-JPEG detection. The rationale behind this choice is that, by optimizing feature dimensionality reduction method, we should be able to obtain better performance than methods that exploit first significant digits as DCT histogram reduction methods [66].

2.4.2 Proposed Method

The proposed pipeline for multiple-JPEG compression classification is depicted in Figure 2.9. During training, a feature vector is extracted from training images. Then, TNMF algorithm is used in order to jointly learn a dictionary for feature reduction and estimate parameters of a supervised classifier. When the system is trained, a new image can be tested. To this purpose, a feature vector is extracted from the image. Features are projected into a reduced dimensionality space using the learned dictionary and eventually a classifier is used to detect the number of compressions. In the following, we provide a detailed analysis of each step of the algorithm and a simplified example of how TNMF dimensionality reduction works.

Feature extraction

In order to start our analysis, we first extract a set of selected features from each image. We opted for the feature extraction pipeline presented in [62], which exploits block-wise DCT histograms of the image. Multiple compression stages are well known to strongly condition the histograms of DCT coefficients, hence justifying our approach. In particular, the set of investigated coefficients includes only the first 9 AC spatial frequencies taken in zig-zag order. Our choice comes from the more regular trend of lower frequencies coefficients and from the reduced statistics of higher components,

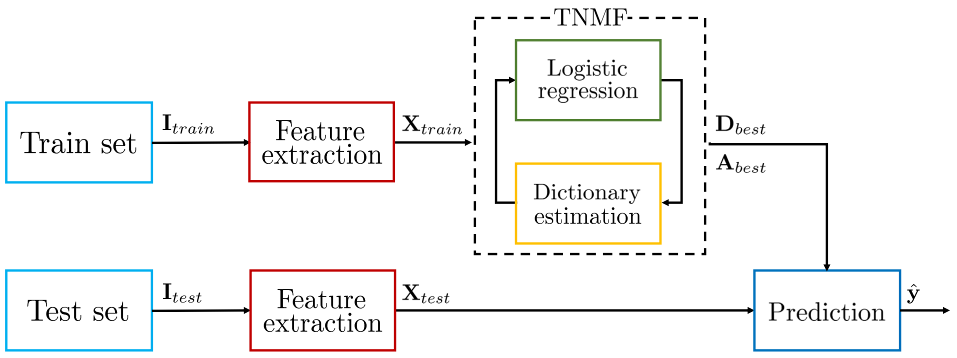


Figure 2.9: *Schematic representation of proposed pipeline.*

which are often quantized to zero [66]. For what concerns the histograms, we select only the first 21 central bins for each DCT band, ending up with a feature vector $\mathbf{x} \in \mathbb{R}_+^n$ of n elements per image, with $n = 189$. Notice that $\mathbf{x}$ assumes non-negative values only.

TNMF Training

Since we aim at classifying multiple compressed images, we propose to exploit multinomial logistic regression, in a one-vs-rest implementation. This means that the multi-class classifier is actually composed by four binary classifiers (e.g., one compression vs. others, two compressions vs. others, etc.), and results of these are merged. For each class label, we train the classifier by minimizing the logistic loss function, defined as $l_s = l_s(y_i, \mathbf{A})$, where y_i is the image label and $\mathbf{A} = \{\mathbf{w}, \mathbf{c}\}$ is the parameter configuration related to that class [73].

In particular, we propose to exploit Task-driven Non-negative Matrix Factorization (TNMF), which is capable of finding sparse data representations by learning a dictionary suited to the specific task of classification [67]. TNMF model allows to learn the task-driven dictionary and the classifier parameters in a joint iterative fashion. More specifically, we estimate a classifier which is optimized with respect to standard logistic regressor, thanks to a particular representation of input data: DCT features extracted from images are projected on a dictionary that is actually tailored to our multinomial classification task. The algorithm works iteratively, alternating the updating of classifier and dictionary, until a fixed number of iterations is achieved. It follows an exhaustive illustration of the method.

a) Feature reduction. At each iteration t, TNMF starts with feature reduction. First of all, we select as dictionary the one from the previous

iteration, hence $\mathbf{D}_t = \mathbf{D}_{t-1}$, with $\mathbf{D}_t \in \mathbb{R}_+^{n \times p}$. Given a training vector $\mathbf{x}_i$ generated from image $\mathbf{I}_i$, TNMF model considers the optimal projections of data points on the dictionary, with the constraints that all the elements of $\mathbf{x}_i$, $\mathbf{D}_t$ and the obtained projections are non-negative. Notice that p corresponds to the desired size of reduced features, n is the input feature size, thus $p < n$.
Typically, the problem is formulated as follows:

$$\mathbf{h}_{i,t}(\mathbf{D}_t) = \arg \min_{\mathbf{h} \in \mathbb{R}_+^p} \; \|\mathbf{x}_i - \mathbf{D}_t\,\mathbf{h}\|_2^2 \; + \; \lambda_1 ||\mathbf{h}||_1 \; + \; \lambda_2 ||\mathbf{h}||_2^2 \,. \tag{2.7}$$

$\mathbf{h}_{i,t}(\mathbf{D}_t) \in \mathbb{R}_+^p$ is the estimated projection, λ_1 and λ_2 are regularization penalty terms, in order to impose sparsity (ℓ_1 norm) and to obtain a strongly convex problem (ℓ_2 norm) hence guaranteeing a unique solution. Eq. (2.7) can be solved with standard techniques available in the literature as shall be cleared in the experimental results section.

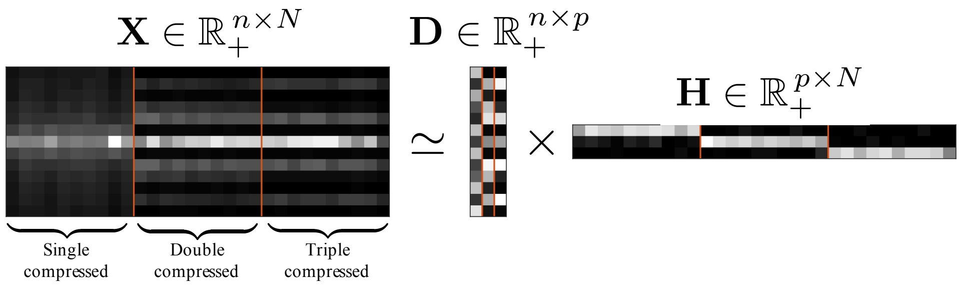
b) Classifier updating. Once we have defined the optimal projections, we can use them to update the classifier, associating each vector $\mathbf{h}_{i,t}$ to its related class label y_i. This operation is performed by the minimization of the expected value of loss function l_s over the entire training set:

$$\mathbf{w}_t,\, \mathbf{c}_t = \arg \min_{\mathbf{w},\mathbf{c}} \; \mathbb{E}_{y_i,\mathbf{x}_i}[l_s(y_i, \mathbf{w}, \mathbf{c}, \mathbf{h}_{i,t}(\mathbf{D}_t))] \; + \; \nu ||\mathbf{w}||_2^2 \,, \tag{2.8}$$

where ν is the penalty term of the regularizer, introduced to prevent overfitting in the classifier. In the proposed framework, minimization is solved by means of the L-BFGS iterative algorithm [74]. In other words, this step consists in training the multi-class logistic regressor exploiting projected training data samples $\mathbf{h}_{i,t}$ and their labels.

c) Dictionary updating. At the end of each iteration t, the dictionary must be updated considering the trained logistic regressor at step *(b)*, thus obtaining $\mathbf{D}_t$ to be used during next iteration $t + 1$. This is done through the minimization of loss function (2.8) with respect to the dictionary $\mathbf{D}_t$. In particular, we update the dictionary by means of stochastic gradient descent, evaluating the function in each training sample $\mathbf{x}_i$ and minimizing in an iterative manner. In order to perform this task, we have to re-evaluate the sparse representation of each training data sample, $\mathbf{h}_{i,t}(\mathbf{D}_t)$, which depends on the dictionary $\mathbf{D}_t$ estimated from already analyzed samples ($\mathbf{x}_j$, $j < i$). To be more specific, the minimization is performed in two sequential steps:

- We exploit stochastic gradient descent by calculating the gradient with respect to $\mathbf{h}_{i,t}(\mathbf{D}_t)$. Since we work with sparse representations of data, we compute the active set $\mathcal{S}$ by selecting only the indexes $\in \{1, ..., p\}$ for



$\mathbf{X} \in \mathbb{R}_+^{n \times N}$
$\mathbf{D} \in \mathbb{R}_+^{n \times p}$
$\mathbf{H} \in \mathbb{R}_+^{p \times N}$
$\simeq$
$\times$
Single compressed
Double compressed
Triple compressed

Figure 2.10: *Simple TNMF example:* $\mathbf{X}$ *is the matrix of data, specifically* N *is the total amount of training samples. We can exploit TNMF to approximate* $\mathbf{X}$ *as the product of the dictionary* $\mathbf{D}$ *and the matrix* $\mathbf{H}$ *containing in its columns the optimal projections of* $\mathbf{X}$ *on* $\mathbf{D}$.

which vector $\mathbf{h}_{i,t}(\mathbf{D}_t) \neq 0$. For the sake of notation, we introduce the variable $\mathbf{g}_{i,t}$, defined as:

$$\mathbf{g}_{i,t} = ([\mathbf{D}_t^\top]_\mathcal{S}[\mathbf{D}_t]_\mathcal{S} + \lambda_2 \mathbf{I}_{|\mathcal{S}|})^{-1}[\nabla_{\mathbf{h}_{i,t(\mathbf{D}_t)}} l_s(y, \mathbf{A}_t, \mathbf{h}_{i,t}(\mathbf{D}_t))]_\mathcal{S}$$

where symbol $[\cdot]_\mathcal{S}$ represents the projection on the active set, and $|\mathcal{S}|$ is the cardinality of $\mathcal{S}$.

- It follows a projection of $\mathbf{g}_{i,t}$ over the dictionary space, leading to this updating formulation:

$$\mathbf{D}_t = \mathbf{D}_t \; - \; \rho_t(-\mathbf{D}_t \, \mathbf{g}_{i,t} \, \mathbf{h}_{i,t}^\top(\mathbf{D}_t) \; + \; (\mathbf{x}_i - \mathbf{D}_t \, \mathbf{h}_{i,t}(\mathbf{D}_t)) \, \mathbf{g}_{i,t}) \, .$$

In order to impose the non-negativity of each dictionary element, we select $\epsilon = 10^{-7}$ as floor value in case of negative entries of $\mathbf{D}_t$. The learning rate ρ_t is chosen with the same heuristic criterion proposed in [67]: we select it as $\min(\rho, \rho \cdot n_{iter}/(10t))$, being ρ a parameter to set and n_{iter} the number of iterations. Given the conspicuous theoretical baggage of TNMF, we skip all the formal derivations, addressing the interested reader to [67] for a thorough explanation.

Following the typical framework of dictionary learning problems, we adopt a validation strategy for selecting the best dictionary and classifier. More specifically, we split our data in training and validation set, evaluating the classification accuracy on validation set at each iteration t, and electing as best dictionary the matrix $\mathbf{D}_t$ which returns the best accuracy. Algorithm 2.1 reports the pseudo-code of TNMF method.

TNMF Testing

Once we estimate the best combination of dictionary and classifier, we are ready for test phase. Given any new image $\mathbf{I}_{test}$, we compute DCT his-

2.1 TNMF Training $(\mathbf{y}, \mathbf{X}, p, n_{iter}, \lambda_1, \lambda_2, \nu) \rightarrow \mathbf{D}_{best}, \mathbf{A}_{best}$

```
Initialize dictionary $\mathbf{D}_0$
Split training and validation sets:
$\mathbf{y}_{train},\, \mathbf{y}_{val},\, \mathbf{X}_{train},\, \mathbf{X}_{val} \leftarrow \mathbf{y},\, \mathbf{X}$
for $t = 1, \dots, n_{iter}$
    Initialize dictionary: $\mathbf{D}_t = \mathbf{D}_{t-1}$
    Feature reduction: $\mathbf{h}_t(\mathbf{D}_t) \leftarrow \mathbf{D}_t,\, \mathbf{X}_{train}$
    Classifier updating: $\mathbf{A}_t \leftarrow \mathbf{h}_t(\mathbf{D}_t),\, \mathbf{y}_{train}$
    Validation accuracy: acc $\leftarrow \mathbf{D}_t,\, \mathbf{A}_t,\, \mathbf{y}_{val},\, \mathbf{X}_{val}$
    $\mathbf{D}_{best},\, \mathbf{A}_{best} \leftarrow \mathrm{acc}_{\max}\{\mathbf{D}_t,\, \mathbf{A}_t\}$
    for $i = 1, \dots, N$
        Single feature extraction: $\mathbf{h}_{i,t}(\mathbf{D}_t) \leftarrow \mathbf{D}_t,\, \mathbf{x}_{train_i}$
        Active set: $\mathcal{S} \leftarrow$ indexes $\in \{1, \dots, p\} : \mathbf{h}_{i,t}(\mathbf{D}_t) \neq 0$
        Update learning rate: $\rho_t \leftarrow \min(\rho, \rho\frac{t_0}{t})$
        Update dictionary:
        $\mathbf{D}_t = \mathbf{D}_t - \rho_t(-\mathbf{D}_t\, \mathbf{g}_{i,t}\, \mathbf{h}_{i,t}^\top(\mathbf{D}_t) + \dots$
                $+ (\mathbf{x}_i - \mathbf{D}_t\, \mathbf{h}_{i,t}(\mathbf{D}_t))\, \mathbf{g}_{i,t})$
        Impose non-negativity: $\mathbf{D}_t(\mathbf{D}_t < 0) = \epsilon$
    end
end
```

tograms to obtain feature vector $\mathbf{x}_{test}$. By considering the best validation dictionary, $\mathbf{D}_{best}$, we apply (2.7) to project $\mathbf{x}_{test}$ and obtain the reduced feature vector $\mathbf{h}_{test}$. Finally, we feed $\mathbf{h}_{test}$ to the logistic regressor using the best validation configuration $\mathbf{A}_{best}$ in order to perform label prediction, as in a typical classification problem.

TNMF Training: a simplified example

For the sake of simplicity and data visualization, let us consider a simplified problem consisting of a small dataset of images compressed up to three times. From each image we extract feature $\mathbf{x}$ only considering the 7-th DCT frequency, picking the first central 13 bins. Selecting as reduced feature size $p = 3$, we leverage TNMF training algorithm to find a dictionary for representing our data. In particular, as we are working in a simplified scenario aiming at a ternary classification (discriminating up to three JPEG compressions), a good feature reduction method should enable ternary classification in the reduced space. Figure 2.10 depicts the results of dictionary learning through TNMF: we are actually able to estimate a dictionary, associating reduced features to original input data (i.e., classification result is clear just by looking at projected features). Notice that matrix $\mathbf{X}$ illustrates quite well the effects of multiple quantization steps: the more the compression stages, the lower the density of the histogram bins.

2.4.3 Experimental results

Datasets generation

Following the procedure depicted in [66], we build three datasets starting from 1338 images from UCID [75] (384×512 pixels). Given a final quality factor $\text{QF}_f \in \{75, 80, 90\}$, we compress each grayscale image up to 4 times. The intermediate QF at compression step $i < f$ is randomly chosen in the interval $[\text{QF}_{i+1} - 12,\ \text{QF}_{i+1} - 5] \cup [\text{QF}_{i+1} + 5,\ \text{QF}_{i+1} + 12]$ to ensure that QF_i differs from QF_{i+1}. We refer to these datasets as $\mathcal{D}^U_{75}$, $\mathcal{D}^U_{80}$, $\mathcal{D}^U_{90}$, each of them with $4 \times 1338 = 5352$ images. In order to test our approach on a larger scale, we build three further datasets starting with 4000 grayscale images from RAISE database [76]. Due to the large dimensions of these images, we previously center-crop them to 512×512 pixels and then apply the aforementioned compression pipeline. We obtain $\mathcal{D}^R_{75}$, $\mathcal{D}^R_{80}$, $\mathcal{D}^R_{90}$, each of them with $4 \times 4000 = 16000$ images.

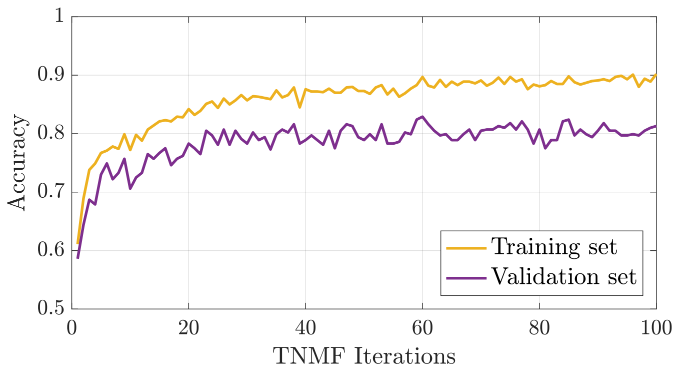
TNMF parameters

We follow a common train-validation-test approach, using 70% of each dataset images for training and the remaining for testing. Training set is further divided in training and validation, following a $90\% - 10\%$ partition. In particular, as recommended in [67], we initialize the dictionary $\mathbf{D}_0$ by the unsupervised formulation of the problem, leveraging the SPAMS toolbox for computations [77]. The size of $\mathbf{D}_0$ has been chosen as trade off between result quality and computational cost: we set p as the 30% of the DCT length, hence drawing a dictionary $\in \mathbb{R}_+^{189 \times 57}$.

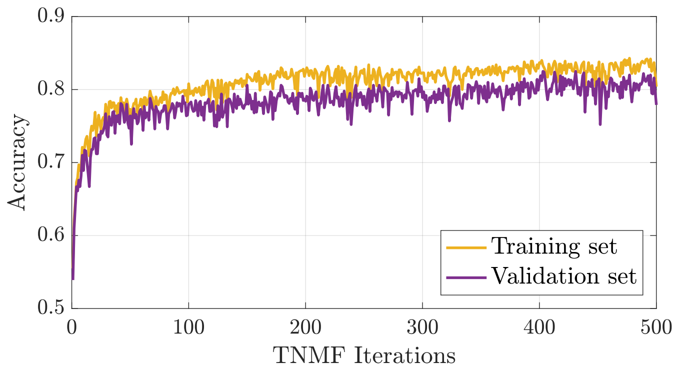
Moreover, in order to improve the convergence speed of the training phase, the proposed method works with a minibatch strategy for the stochastic gradient descent. This basically takes into consideration $n_{batch} > 1$ training samples at each iteration of the inner loop, instead of a single one [77].

Due to the large amount of parameters of TNMF algorithm, we select some specific values for tuning (i.e., iterations $\in \{50, 100, 200, 500\}$, batch size $\in \{200, 400\}$, $\rho \in \{0.001, 0.005, 0.01\}$, $\lambda_1 \in \{0.01, 0.1, 0.5\}$) and perform a grid search in order to obtain the best accuracy on the validation set. In particular, being TNMF an iterative algorithm, number of iterations has a severe impact on validation accuracy, thus we explore different values until convergence.

For what concerns the remaining parameters, we set $\lambda_2 = 0$ drawing the idea from [67], even though $\lambda_2 > 0$ would be necessary for the differentiability of (2.7). This has proven to get satisfactory results in most



(a)



(b)

Figure 2.11: *Classification accuracy of TNMF algorithm. (a) UCID Dataset. (b) RAISE Dataset.*

experiments. The penalty weight in (2.8) is left untouched with respect to the standard formulation of logistic regressor, hence $\nu = 1$.

Comparison with state-of-the-art

At first glance, we notice that the algorithm needs more iterations on RAISE-derived datasets than on the UCID ones. This is probably due to the huge difference in terms of dataset size. In this vein, Figure 2.11 depicts the temporal evolution of TNMF accuracy, evaluated for training and validation sets on $\mathcal{D}_{75}^{U}$ and $\mathcal{D}_{75}^{R}$. Notice that, if on $\mathcal{D}_{75}^{U}$ we achieve convergence in at most 100 iterations, $\mathcal{D}_{75}^{R}$ requires more than 200 iterations. For the sake of brevity, we are showing results for the specific combination of parameters which yields the best validation accuracy, and we will stick to this approach from now on. Nonetheless, we performed a comprehensive investigation on

Table 2.1: *Mean test accuracies over 4 classes for proposed TNMF method, classifier in [66], and logistic regressor (LR) without feature reduction. Best results in bold.*

Accuracy	TNMF	[66]	LR
$\mathcal{D}^U_{75}$	**0.78**	0.73	0.64
$\mathcal{D}^U_{80}$	**0.82**	0.75	0.65
$\mathcal{D}^U_{90}$	**0.87**	0.80	0.75

(a)

Accuracy	TNMF	[66]	LR
$\mathcal{D}^R_{75}$	**0.80**	0.76	0.64
$\mathcal{D}^R_{80}$	**0.81**	0.75	0.64
$\mathcal{D}^R_{90}$	**0.87**	0.83	0.74

(b)

Table 2.2: *Confusion matrices for $\mathcal{D}^R_{80}$, $\mathcal{D}^R_{90}$. Top: proposed method. Bottom: method in [66]. The highest accuracy among the two methods for a given compression step and dataset is highlighted in yellow.*

$\mathcal{D}^R_{80}$	1	2	3	4
1	**0.980**	0.004	0.015	0.001
2	0.009	**0.850**	0.068	0.073
3	0.079	0.119	**0.711**	0.091
4	0.005	0.158	0.117	**0.720**

$\mathcal{D}^R_{90}$	1	2	3	4
1	**0.997**	0.002	0.001	0.000
2	0.005	**0.952**	0.022	0.021
3	0.022	0.048	**0.833**	0.097
4	0.000	0.126	0.175	**0.699**

$\mathcal{D}^R_{80}$	1	2	3	4
1	**0.999**	0.000	0.000	0.001
2	0.025	**0.960**	0.012	0.003
3	0.182	0.234	**0.523**	0.061
4	0.094	0.275	0.109	**0.522**

$\mathcal{D}^R_{90}$	1	2	3	4
1	**1.000**	0.000	0.000	0.000
2	0.130	**0.863**	0.004	0.003
3	0.029	0.088	**0.828**	0.055
4	0.020	0.179	0.165	**0.636**

test results for all the parameter configurations, obtaining puny variations among them (standard deviation of test accuracy < 0.01).

Tables 2.1 and 2.2 show results of the test phase, in terms of mean accuracies and confusion matrices. Specifically, we compare our strategy to [66] (i.e., the only algorithm that deals with four JPEG compressions to the best of our knowledge) and to standard logistic regression without the feature reduction step. This last experiment is used to study the actual positive effect of dictionary projection.

For what concerns the accuracy, our solution is able to go beyond the previously proposed method, since the overall average accuracy (considering all the datasets) is 5.5 percentage points above the mean accuracy of [66]. Regarding the confusion matrices, our results are more accurate than [66] in detection of classes 3 and 4. Indeed, for datasets $\mathcal{D}^R_{75}$ and $\mathcal{D}^R_{80}$ the diagonal terms corresponding to classes 3 and 4 present an average gap of $+0.15$ with respect to state-of-the-art, which achieves acceptable multi-classification outcomes especially when QF_f increases up to 90. Concerning the other classes, our results are comparable to [66] for single compressed images, while the detection of double compressed is slightly overwhelmed, probably dictated by a better accuracy of further compressions.

Robustness

In order to preliminary test the method's resilience to editing operations in between JPEG compressions, we applied our detector trained on $\mathcal{D}_{90}^{R}$ to images that randomly underwent either blurring or gamma correction in addition to compression. This campaign shows that if a single transformation is performed, accuracy drops approximately by 10%. The main effect of the transformations is to hide the previously applied JPEG compression. This paves the way to thrilling future research scenarios.

2.5 Conclusions

In this chapter we present methods for source device identification on images. Contrarily to state-of-the-art solutions which leverage statistical approaches to match images with the correct source camera, we exploit convolutional neural networks. The main motivation of our research is to find efficient solutions when memory storage and computation time are a concern (e.g., in situations where a huge device collection must be analyzed).

Specifically, we make use of the characteristic fingerprint of devices (i.e., the PRNU) and the image noise residuals to train a neural network able to extract subtle sensor traces of each device, thus enabling to link images to the correct source. To this purpose, we propose diverse configurations of network architectures, training strategies and loss functions. Results show that CNN-based approach can exceed state-of-the-art performances, both in terms of accuracy and time.

During our experimental campaign, we test the proposed methodology on images compressed just once. However, an important portion of digital images circulating over the Internet usually undergoes several compressions due to various processing and sharing operations. Actually, being able to infer the number of compressions an image underwent can provide the analyst an instrument to assess the integrity of digital content. This may be extremely helpful as primary step before to start with the identification of source device.

In this vein, we propose a novel method for detecting multiple JPEG compressions, considering up to four coding steps. Our approach takes advantage of Task-driven Non-negative Matrix Factorization (TNMF) model, both for feature reduction and for classification, through a joint iterative estimation of dictionary and classifier. We extensively test several setups, taking into account different datasets and quality factors. Experiments show that our method outperforms up to date state-of-the-art in terms of classification accuracy.

CHAPTER 3

Source device identification on videos

In this chapter, we investigate source device identification problem on video sequences. We start showing state-of-the-art solutions which tackle typical problems occurring on video sequences, such as low video frame resolution and strong compression. Then, we thoroughly analyze the issue of stabilization and explain why this technology strongly hinders performances of source identification algorithms on stabilized video sequences.

In this vein, we propose diverse methodologies for facing source device identification on stabilized videos. Specifically, we present two methods to compute the device reference fingerprint exploiting stabilized frames and three different solutions to correctly attribute a query stabilized video to the original source camera. To this purpose, we use coordinate system transformations and global optimization algorithms in order to match each video to the correct device.

Eventually, we show an application of the source device identification problem on compilations of videos created by temporarily aligning diverse video sequences. The goal is to blind detect and localize the splicing points of the query sequence. Precisely, we do not known the amount of combined sequences, their temporal duration and the original source cameras. To face the problem, we extract sensor-based footprints inspired to the device characteristic fingerprint. Our method is validated on both non-stabilized and stabilized videos, thus showing the difficulty of working in the latter scenario.

3.1 State-of-the-art for video-camera attribution problem

As reported in Section 2.2, the most powerful methods for device identification on images rely on camera photo response non-uniformity (PRNU). Whether the source device of an image must be identified, we can resort to cross-correlation measures between a noise residual extracted from the image and the PRNU of the candidate device.

PRNU-based methods have been readily extended to video to accomplish a variety of forensic tasks, e.g., source identification [23], detection of duplicate videos [78] authentication of smartphones [79]. However, extending PRNU-based methods to video sequences is not straightforward, and presents multiple challenges [13]. Indeed, video signals are typically less reliable than images due to their lower resolution, as well as stronger compression.

Gaining robustness against compression is a primary goal of current research, since videos are often uploaded on YouTube [80] or shared through other social networks [39, 81]. In [23] blocking artifacts caused by compression are corrected before evaluating decision statistics. In [82] a confidence weighting scheme is proposed to identify high-frequency areas of the scene, which are discarded to ensure a more reliable PRNU estimation. In [83] video frames are reordered and weighed according to their reliability, given that I-frames enable better PRNU estimate than P-frames. Also, videos delivered on a wireless network suffer from blocking and blurring due to packet losses, and suitable algorithms need to be developed to handle this situation [84].

Therefore, PRNU traces in video sequences tend to be very subtle. This is confirmed by the authors of [23, 83], who propose to consider each video frame as a picture and follow the standard PRNU-based pipeline for image attribution. Their results show that not all video frames can be considered as equally informative (i.e., intra-coded frames typically contain more reliable PRNU information).

Interesting alternative approaches have been proposed by [12, 13]. The authors suggest to estimate camera PRNU from images as reported in (2.1), and use it for video attribution. However, the authors cannot compare image PRNU with video noise directly, as video resolution is typically lower than image one [23]. In order to adapt the sensor size to the video recording area, they propose a strategy that searches for a correct scale and crop transformation to match image PRNU and video resolutions.

Another major problem is video stabilization, which applies geometrical transformations (e.g., translations, similarities, homographies, etc.) to acquired video frames in order to compensate for involuntary user's movement [30]. This causes misalignment of individual pixels across frames, preventing a reliable estimation of the PRNU fingerprint. Since modern smartphone cameras adopt video stabilization, and most of the videos uploaded on the Internet come from smartphones, PRNU-based methods may be of little use [29] without suitable corrections.

The first paper addressing this problem [85] dates back to 2011, but it only takes translations into account. In [13], it is more realistically assumed that stabilization is performed using a combination of translation and rotation, which are estimated and compensated on I-frames only. In the same work, it is also proposed to perform video camera attribution using a set of images from the same camera. This idea is further developed in [12] where a hybrid sensor pattern noise analysis is carried out to handle the problem of video stabilization. Specifically, the reference PRNU is estimated using only still images, whereas query videos are mapped to the image domain by compensating for possible scale and translation transformations.

In the following, we enter more in details of video stabilization technology, in order to clarify why it can impede the performances of standard PRNU-based methods.

3.1.1 In-camera video motion stabilization

Since a significant percentage of videos shared online is captured by amateur users, which are usually not equipped by professional stabilization tools, these videos often suffer from camera-shake. Motion is principally induced by the hand-held capturing process, but can also be due to other movements of the users that may walk or even run while recording. As a consequence, plenty of strategies to perform the stabilization of a video (directly on the recording camera or off-line) have been proposed [30, 86–90].

Video motion stabilization methods allow to improve the quality of the recorded videos, making each sequence appearing as if it were recorded from a stable camera moving along a smooth path. In particular, these systems are able to detect and correct high frequency jitter artifacts, low frequency artifacts, rolling shutter wobbles, foreground motion, poor lighting, and scene cuts [30].

Among the most recent state-of-the-art approaches, the authors of [30] propose to perform video stabilization by fitting the original 2D camera path with linear motion models, characterized by a different amount of degrees of freedom (DOF). Whenever these models are considered to be valid for the considered frame-pair motion, the original path is transformed according to the model and a smooth camera path is generated. Frames are then warped on this new path by applying a set of pixel-wise transformations.

The easiest motion model describes only translations, hence 2 DOF. This can be represented by means of a pixel coordinates transformation matrix $\mathbf{T}_2$, defined as

$$\mathbf{T}_2 = \begin{pmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \end{pmatrix}, \tag{3.1}$$

where c_x and c_y are the magnitude of translation of the camera along the horizontal and vertical axes, respectively. Alternatively, in order to detect also rotation and uniform scaling between frames, the similarity model including 4 DOF can be used. The matrix describing the motion relationship between frames is

$$\mathbf{T}_4 = \begin{pmatrix} s \cdot \cos\alpha & -s \cdot \sin\alpha & c_x \\ s \cdot \sin\alpha & s \cdot \cos\alpha & c_y \end{pmatrix}, \tag{3.2}$$

being s and α the scaling factor and rotation angle, respectively. More complex homographic models can also be considered if perspective distortions have to be recovered. However, not every model can efficiently represent the motion between two frames, and the application of an incorrect motion model introduces distortions in the stabilized video. As an example, whenever the model is invalid, translations and similarities inject additional shaking in the estimated path, whereas the homographic models result in perspective warping errors. Moreover, the higher the complexity of the used model, the higher the probability of wrongly estimating it, potentially leading to temporal instability of the generated path [87, 91].

In the light of this, stabilization methods usually perform a first step to delete the shake due to similarity and lower DOF motions, without taking into account higher DOF. Then, any residual motion can be additionally corrected exploiting the homographic models if needed [30]. This two-step approach comes in handy whenever computational complexity is an issue. Indeed, if stabilization is performed on mobile devices, a single step can be used.

A consequence of motion stabilization on a video sequence is that two pixels sharing the same geometrical coordinates on two different frames

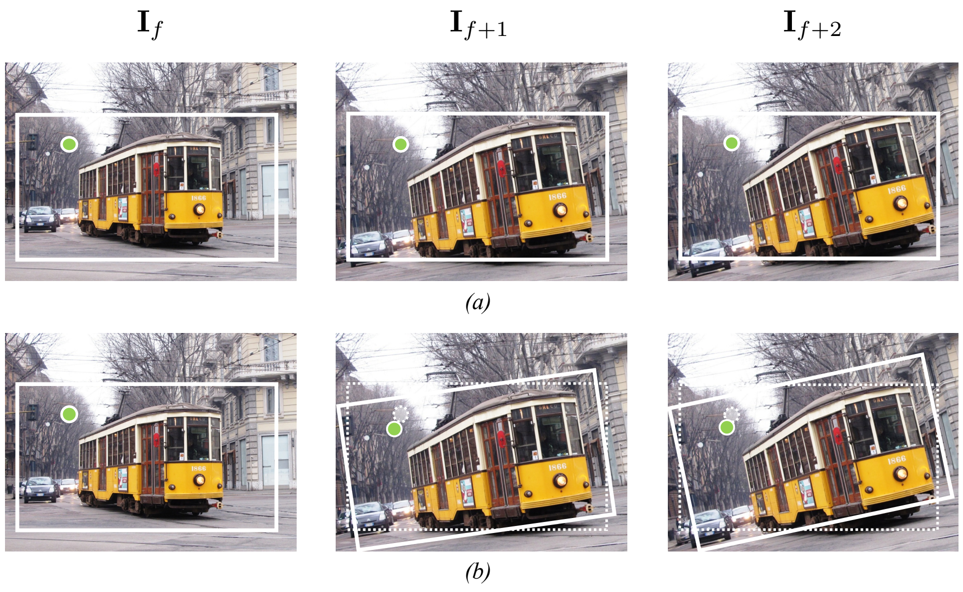


Figure 3.1: *(a) In absence of stabilization, the pixel coordinates (green circle) do not change from frame to frame; (b) In presence of stabilization, recording sensor area can be slightly shifted, scaled and rotated, hence the pixel coordinates (green circle) can change with respect to the original ones (shown in dashed line).*

may have been acquired with different portions of the camera sensor due to the introduced geometrical transformations. For the sake of clarity, Figure 3.1 reports three adjacent frames of a video. The area inside the white box highlights the final scene depicted on the recorded video by the device. In the first row, the depicted scene in absence of stabilization: selecting a pixel inside the recording area, its coordinates maintain fixed during capture. Whenever stabilization is present (second row), in order to generate a stable camera path, each pixel can actually vary its coordinates during recording. As shall be clear in the next section, this is a problem for PRNU-based video camera attribution.

3.1.2 Pixel transformations introduced by video stabilization

As reported above, solving the camera attribution problem on videos is far from being an easy task. First, video sequences typically have a different resolution with respect to PRNU. Actually, this issue could be solved by searching the correct scale and crop transformation to match the PRNU and video frame resolutions. However, if video stabilization is used, additional transformations may have been applied as well.

In the light of these considerations, we propose to exploit a 4-parameter linear model to describe the relation between image and video frame domains, modeling both the operation that shrinks the recording area and the stabilization counteracting global frame shake [87,90]. The considered linear model consists in a 2D similarity transformation $\mathscr{T}$ resulting from the combination of two geometric transformations: (i) the first, $\mathscr{T}_{\mathsf{iv}}$, depicting the effect of sensor area shrinkage into video resolution (represented by the matrix $\mathbf{T}_{\mathsf{iv}}$); (ii) the second, $\mathscr{T}_{\mathsf{v}}$, describing the video frame stabilization process (depicted by the matrix $\mathbf{T}_{\mathsf{v}}$). In particular,

$$\mathbf{T}_{\mathsf{iv}} = \begin{pmatrix} s_{\mathsf{iv}} & 0 & c_{\mathsf{iv}_x} \\ 0 & s_{\mathsf{iv}} & c_{\mathsf{iv}_y} \end{pmatrix}, \tag{3.3}$$

and

$$\mathbf{T}_{\mathsf{v}} = \begin{pmatrix} s_{\mathsf{v}} \cdot \cos\alpha_{\mathsf{v}} & -s_{\mathsf{v}} \cdot \sin\alpha_{\mathsf{v}} & c_{\mathsf{v}_x} \\ s_{\mathsf{v}} \cdot \sin\alpha_{\mathsf{v}} & s_{\mathsf{v}} \cdot \cos\alpha_{\mathsf{v}} & c_{\mathsf{v}_y} \end{pmatrix}. \tag{3.4}$$

$\mathbf{T}_{\mathsf{iv}}$ models the scaling and cropping operations performed in order to map high resolution PRNU into video resolution [12,13]: s_{iv} describes the scale and $\mathbf{c}_{\mathsf{iv}} = (c_{\mathsf{iv}_x}, c_{\mathsf{iv}_y})$ the translation along x and y axes, respectively.

Notice that the transformation $\mathscr{T}_{\mathsf{iv}}$ depends on video and image resolutions, which may actually assume different values for the same device. For this reason, $\mathscr{T}_{\mathsf{iv}}$ is not necessarily unique per device. The second matrix $\mathbf{T}_{\mathsf{v}}$ models the additional geometric transformation resulting from video stabilization: s_{v} describes the scale, α_{v} the rotation, and vector $\mathbf{c}_{\mathsf{v}}$ the shifts. The transformation of image to frame domain can be modeled as $\mathcal{I}_f = \mathscr{T}(\mathcal{I})$, being $\mathcal{I}$ the image space and $\mathcal{I}_f$ the space related to video frames.

Concerning non-stabilized videos, the relations expressed by (3.3) and (3.4) can be further simplified by noticing that $\mathbf{T}_{\mathsf{v}}$ is reasonably an identity matrix. As a matter of fact, there would be no reason to change scale, rotation angle and shift between frames when recording a non-stabilized video. Moreover, as reported in [12], all video frames recorded with the same resolution by a unique non-stabilized device are affected by equal scaling and shift factors, being these parameters probably fixed by the device firmware specifications. Therefore, the image to video frame transformation reduces to $\mathscr{T} = \mathscr{T}_{\mathsf{iv}}$.

When video stabilization is used, each frame $\mathbf{I}_f$ in the sequence experiences its own scale, rotation and translation. The overall similarity is frame-specific, thus can be defined as $\mathscr{T}_f$, which can be modeled by

$$\mathbf{T}_f = \begin{pmatrix} s_f \cdot \cos\alpha_f & -s_f \cdot \sin\alpha_f & c_{x_f} \\ s_f \cdot \sin\alpha_f & s_f \cdot \cos\alpha_f & c_{y_f} \end{pmatrix}, \tag{3.5}$$

where the tuple $\{s_f, \alpha_f, \mathbf{c}_f\}$ results from the entries of the matrix product $\mathbf{T}_{\mathsf{v}}\mathbf{T}_{\mathsf{iv}}$. Specifically,

- $s_f = s_{\mathsf{iv}} \cdot s_{\mathsf{v}}(f)$;
- $\alpha_f = \alpha_{\mathsf{v}}(f)$;
- $c_{x_f} = c_{\mathsf{iv}_x} \cdot s_{\mathsf{v}}(f) \cdot \cos\alpha_{\mathsf{v}}(f) - c_{\mathsf{iv}_y} \cdot s_{\mathsf{v}}(f) \cdot \sin\alpha_{\mathsf{v}}(f) + c_{\mathsf{v}_x}(f)$;
- $c_{y_f} = c_{\mathsf{iv}_x} \cdot s_{\mathsf{v}}(f) \cdot \sin\alpha_{\mathsf{v}}(f) + c_{\mathsf{iv}_y} \cdot s_{\mathsf{v}}(f) \cdot \cos\alpha_{\mathsf{v}}(f) + c_{\mathsf{v}_y}(f)$.

All these transformations should be taken into account when dealing with PRNU-related problems.

3.2 Source device identification on stabilized video sequences

Following the previous considerations, in this section we focus on the problem of video source identification in the challenging scenario of in-camera stabilized video sequences. In order to solve this problem, we exploit PRNU-based traces and split the workflow into two separate steps:

- given some multimedia content acquired with a device, estimate its PRNU-based fingerprint;
- given this fingerprint and a video query, detect whether the video comes from the camera under analysis.

Concerning the first step, the primary goal is finding a good estimation of the camera fingerprint in the video resolution domain. To this purpose, it is reasonable to consider three main strategies, depending on the data owned by the analyst:

- exploiting only images shot by the camera to estimate $\mathbf{K}$, then transform it into the video frame domain, given that conversion parameters reported in (3.5) are known for each frame;
- exploiting both images and videos shot by the camera, without knowing the conversion parameters;
- exploiting only videos recorded by the camera.

To the best of our knowledge, the first case is not realistic as the warping image-to-frame parameters are not apriori known neither can be reported in the literature. Therefore, we focus on the other two scenarios. The pipeline of the proposed method is depicted in Figure 3.2. In the following, we present the proposed strategies for fingerprint estimation and video source attribution, discussing the main intuitions behind the approaches.

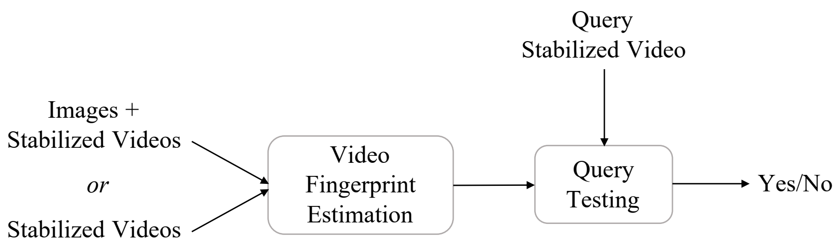


Figure 3.2: *Pipeline of the proposed method. Initially, the device signature is estimated, using images and videos captured by the camera, or videos only. Then, each video query is tested and eventually attributed or not to the camera.*

3.2.1 Reference video fingerprint estimation from images and videos

In the first scenario, the analyst has a set of images and videos recorded with the same camera. We propose a pipeline composed by four main steps:

1. Estimate the device PRNU $\mathbf{K}$ from the available set of pictures applying the definition provided in (2.1).
2. Estimate the image-to-frame transformation parameters $\{s_f, \alpha_f, \mathbf{c}_f\}$ defined in (3.5), by solving an iterative maximization problem for each analyzed frame.
3. Exploit the parameters $\{s_f, \alpha_f, \mathbf{c}_f\}$ for estimating the parameters s_{iv} and $\mathbf{c}_{\mathsf{iv}}$ of the transformation $\mathscr{T}_{\mathsf{iv}}$.
4. Estimate the device fingerprint $\mathbf{K}_{\mathsf{iv}}$ in video domain, by warping $\mathbf{K}$ with the estimated s_{iv} and $\mathbf{c}_{\mathsf{iv}}$, hence $\mathbf{K}_{\mathsf{iv}} = \mathscr{T}_{\mathsf{iv}}(\mathbf{K})$.

In other words, we propose to use as video fingerprint a transformed PRNU, downsampled to the scale of video frame's resolution. This fingerprint is denoted as $\mathbf{K}_{\mathsf{iv}}$ since it is computed using images and then it is converted into the video resolution. We propose this pipeline driven by the following observations:

- Images are often acquired at higher resolution and with better coding quality than videos, thus they typically contain more reliable device fingerprint information.
- Downsampling the image PRNU $\mathbf{K}$ to the video frame's scale requires less computational power than upsampling video frames to the scale of $\mathbf{K}$ (e.g., smaller matrices to fit into memory, PCE correlation computed on less samples, etc.).

- It has been shown in [11] that PRNU downscaling of a factor up to 2 does not significantly hinder camera attribution performance, which is good news considering that image resolution is rarely twice that of a video.

In order to register the image PRNU into video frame domain, we first estimate the image-to-frame conversion tuple $\{s_f, \alpha_f, \mathbf{c}_f\}$ for each selected frame. To infer these parameters, we search for the similarity transformation $\mathcal{T}_f$ that maximizes the PCE correlation between the transformed version of $\mathbf{K}$ and frame residuals $\mathbf{W}_f$, extracted from frames $\mathbf{I}_f$ belonging to a set of selected frames. Formally, we select reasonable search ranges $\mathcal{S}$, $\mathcal{A}$, $\mathcal{C}$ for scale, rotation angle and shift, respectively. Then, we estimate $\{s_f, \alpha_f, \mathbf{c}_f\}$ related to each frame $\mathbf{I}_f$ by solving the maximization problem

$$s_f, \alpha_f, \mathbf{c}_f = \underset{s \in \mathcal{S}, \alpha \in \mathcal{A}, \mathbf{c} \in \mathcal{C}}{\arg\max} \ \mathrm{PCE}(\mathbf{W}_f, \ \mathbf{I}_f \cdot \mathcal{T}(\mathbf{K})). \tag{3.6}$$

If frames come from a *non-stabilized* video, the tuple $\{s_f, \alpha_f, \mathbf{c}_f\}$ is expected to be coherent for all frames in the set. Indeed, cameras do not typically use different portions of the sensor from frame to frame. Moreover, as previously reported, α_f is expected to be zero, as non-stabilized videos are not commonly acquired by rotating the sensor. Therefore, any estimated tuple $\{s_{\mathsf{iv}} = s_f, 0, \mathbf{c}_{\mathsf{iv}} = \mathbf{c}_f\}$ from the selected frame-set can be used for warping the image PRNU into the video domain by means of the transformation $\mathcal{T}_{\mathsf{iv}}$.

If frames come from *stabilized* video sequences, $\{s_f, \alpha_f, \mathbf{c}_f\}$ can vary from frame to frame, as each frame is (almost) independently warped based on the content to stabilize. However, if the considered video does not contain strongly textured areas (flat scenes are always suggested for PRNU extraction [10]) and it is not characterized by excessive device shaking, (typically true for videos to be pleasant at visual inspection), we can assume that the set $\{s_f, \theta_f, \mathbf{c}_f\}$ only slightly changes from frame to frame, oscillating around the true set $\{s_{\mathsf{iv}}, 0, \mathbf{c}_{\mathsf{iv}}\}$, which models the shrinkage of recording area and is independent from frame stabilization. Furthermore, it is reasonable to assume that the rotation contribution is likely to be very small, and it is almost zero for at least a small set of frames. Indeed, the captured scene should not look rotated to video viewers.

In order to select a unique parameter set $\{s_{\mathsf{iv}}, 0, \mathbf{c}_{\mathsf{iv}}\}$ for the image-to-video PRNU conversion, we propose to fix $\alpha_f = 0$, and average the estimated s_f and $\mathbf{c}_f$ parameters over the frames with strong PCE. Notice that frames for which rotation parameter is not really zero can be filtered out from our estimate as they will be characterized by low PCE values.

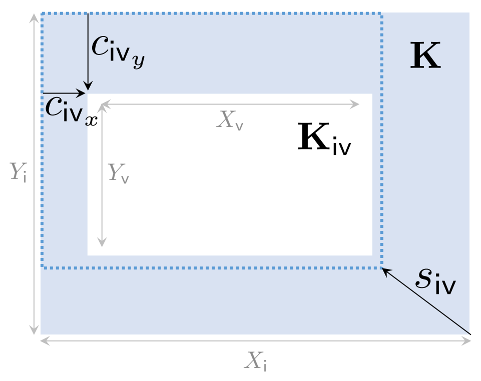


Figure 3.3: *Image PRNU conversion to the video domain. Blue area is the whole sensor with size $X_{\mathsf{i}} \times Y_{\mathsf{i}}$, used for image PRNU, whereas white area represents the video domain fingerprint $\mathbf{K}_{\mathsf{iv}}$, with size $X_{\mathsf{v}} \times Y_{\mathsf{v}}$.*

Formally, we compute

$$p_f = \mathrm{PCE}(\mathbf{W}_f,\ \mathbf{I}_f \cdot \mathscr{T}_f(\mathbf{K})). \tag{3.7}$$

Then, we estimate scale and translation parameters as

$$s_{\mathsf{iv}} = \sum_{f \in \mathcal{F}_{\mathrm{I}}} \frac{s_f}{|\mathcal{F}_{\mathrm{I}}|}, \quad c_{\mathsf{iv}_x} = \sum_{f \in \mathcal{F}_{\mathrm{I}}} \frac{c_{xf}}{|\mathcal{F}_{\mathrm{I}}|}, \quad c_{\mathsf{iv}_y} = \sum_{f \in \mathcal{F}_{\mathrm{I}}} \frac{c_{yf}}{|\mathcal{F}_{\mathrm{I}}|}, \tag{3.8}$$

being $\mathcal{F}_{\mathrm{I}}$ the set of frames for which $p_f > 60$ (i.e., a PCE threshold suggested in [13, 92], which we verified to be a reliable threshold for selecting frames actually matching with the device) and $|\mathcal{F}_{\mathrm{I}}|$ its cardinality.

In order to pass from image to video domain, we apply the similarity transformation $\mathscr{T}_{\mathsf{iv}}$ to the image PRNU, thus obtaining the video fingerprint

$$\mathbf{K}_{\mathsf{iv}} = \mathscr{T}_{\mathsf{iv}}(\mathbf{K}). \tag{3.9}$$

For the sake of clarity, Figure 3.3 depicts the operations done for converting the image PRNU to the video domain. A scale transformation with parameter s_{iv} is performed on $\mathbf{K}$ to shrink the space to a reduced area, then scene is cropped to match video resolution, according to the estimated shift $\mathbf{c}_{\mathsf{iv}}$.

Note that $\mathbf{K}_{\mathsf{iv}}$ can be exploited as device signature for testing the camera attribution problem over a generic video query. Indeed, the resolution of the fingerprint should match the resolution of the query sequence. Moreover, since $\mathbf{K}_{\mathsf{iv}}$ is the result of aligned noise contributions coming from high resolution images, it reasonably contains a very reliable camera model information.

3.2.2 Reference video fingerprint from videos only

The second considered scenario is based on the fact that information about images captured by the device under analysis is not always available. For this reason, we focus on how to estimate the device fingerprint directly from video content. In this setup, we propose a pipeline composed by the following steps:

1. We search for the frame whose residual $\mathbf{W}_f$ correlates well in terms of PCE with the largest number of other frames' residuals.
2. We estimate a candidate video fingerprint $\mathbf{K}_\text{v}$ starting from the selected $\mathbf{W}_f$.
3. We update the video fingerprint $\mathbf{K}_\text{v}$ by iteratively aggregating information from other frames whose noise residuals correlate well with the fingerprint in terms of PCE.

Despite this pipeline seems trivial, the procedure of noise aggregation is not straightforward at all. Actually, due to motion stabilization, if we randomly pick a set of frames and estimate the fingerprint following PRNU estimation as reported in (2.1), noise residuals left by the camera sensor risk to be averaged while misaligned.

To avoid averaging misaligned contributions, noise residuals should be in principle coherently warped one on the other by following a procedure similar to the one proposed in Section 3.2.1. However, sensor noise traces are extremely subtle in video signals. To top it all off, scene content often leaks into frame noise residuals due to the used suboptimal denoising algorithms. These two factors make the estimation of transformation parameters that map a frame noise residual into another one an almost preposterous task.

In order to avoid mistakenly estimating the warping parameters, we make the assumption that a set of video frames affected by similar stabilization transformations $\mathcal{T}_\text{v}$ exists within the available reference videos. This assumption reasonably holds for sequences characterized by low (if any) textured content that does not need to be strongly stabilized (i.e., typical sequences used for PRNU estimation). Under this assumption, we propose an iterative noise residual aggregation method composed by the following steps:

a) Loop over all frames in the set $\mathcal{F}$ of available ones. For each frame $\mathbf{I}_f$, solve the standard camera attribution problem against all other frames $\mathbf{I}_l$. Specifically, compute $\mathrm{PCE}(\mathbf{W}_l, \mathbf{I}_l \cdot \mathbf{W}_f), \forall l, f \in \mathcal{F},\ l \neq f$.

b) Analyze the relative PCE values in search for noise residual's matching. A match is considered if two constraints on the computed PCE are satisfied.

The first constraint is on PCE magnitude. We consider a match only for strictly positive PCE values to avoid strongly uncorrelated frames.

As second constraint, we check the relative shift estimated through PCE (i.e., the position of PCE maximum peak). As a matter of fact, the effect of video stabilization is to scale, rotate and translate the frames one differently from the others, but without introducing visible artifacts on the recorded sequence. For this reason, it is reasonable to assume that a stabilization algorithm does not translate too much one frame with respect to the temporally adjacent ones.

In principle, if frames were not stabilized, the relative shift estimated through PCE should be of $(0, 0)$ pixels, since both frame residuals should be aligned in terms of sensor noise. Conversely, in stabilized videos, the relative alignment can be different from $(0, 0)$. However, under the hypothesis of small translations introduced by stabilization, if the relative alignment is too far from $(0, 0)$ we can attribute it mainly to PCE correlating textured content or additional noise contributions, rather than noise patterns related to the original camera fingerprint.

Therefore, in order to avoid false matching results that do not actually correspond to noise residuals' alignment, we only consider matching residuals if the relative shift is less than (Δ, Δ) pixels ($\Delta = \{5, 10, 20, 30\}$ in our experiments with Full-HD sequences).

c) Select as reference frame $\mathbf{I}_r$ the video frame $\mathbf{I}_f$ that matches with the largest number of frames according to the matching definition provided in step *(b)*.

d) Include the selected frame(s) in the set of frames exploited for the fingerprint estimation, defined as $\mathcal{F}_\mathsf{v}$. For instance, initially $\mathcal{F}_\mathsf{v} = \{\mathbf{I}_r\}$.

e) Update the estimated fingerprint $\mathbf{K}_\mathsf{v}$ by pixel-wise averaging through arithmetic mean all the re-synchronized noise residuals $\mathbf{W}_f, f \in \mathcal{F}_\mathsf{v}$. Thus, at first iteration, the video fingerprint candidate is the residual of frame $\mathbf{I}_r$, namely $\mathbf{K}_\mathsf{v} = \mathbf{W}_r$.

f) Correlate the remaining frames (not in $\mathcal{F}_\mathsf{v}$) with the estimated fingerprint, computing $\mathrm{PCE}(\mathbf{W}_f, \mathbf{I}_f \cdot \mathbf{K}_\mathsf{v}), \forall f \notin \mathcal{F}_\mathsf{v}$.

g) If the PCE related to some frames honors the constraints reported in *(b)*, select these frames and compensate their relative shift misalignment

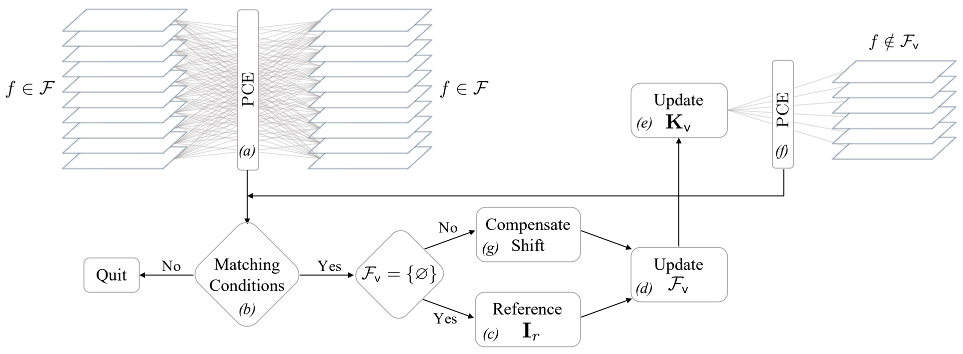


Figure 3.4: *Pipeline of the proposed method for estimating the fingerprint* $\mathbf{K}_v$.

with respect to the estimated fingerprint $\mathbf{K}_v$. Then, go to step *(d)* and continue iterating. Whether no more frame noise residuals match with $\mathbf{K}_v$, stop the iterations.

Eventually, the estimated camera fingerprint for testing the video queries is $\mathbf{K}_v$. For the sake of clarity, Figure 3.4 depicts the proposed pipeline for estimating the device fingerprint $\mathbf{K}_v$.

Notice that there is a big difference between $\mathbf{K}_{iv}$ and $\mathbf{K}_v$. The former is computed from a conspicuous number of high resolution images whose PRNU traces are by default aligned one with respect to the other, and then it is warped into the video resolution. The latter is an aggregation of video noise residuals, which have to be realigned because of stabilization. First of all, frames usually undergo strong compression. Moreover, the set $\mathcal{F}_v$ does not include all the available video frames, but only those satisfying some fixed constraints. Thus, the amount of exploited noise residuals can be reduced, potentially hindering the quality of the fingerprint estimation. For this reason, $\mathbf{K}_{iv}$ can be considered a higher quality estimate of the device video fingerprint compared to $\mathbf{K}_v$.

3.2.3 Testing the video query

Given a device fingerprint $\mathbf{K}_d$ (being either $\mathbf{K}_{iv}$ or $\mathbf{K}_v$) and a video to be attributed, we propose to test a set $\mathcal{F}$ of frames belonging to the sequence following a similar procedure to the standard PCE-based method. Specifically, we estimate the warping configuration that maximizes the PCE between each frame and the transformed fingerprint $\mathbf{K}_d$. In this way, even if the fingerprint has already been registered into video domain, we can compensate for the additional stabilization deviations introduced on query frames. In order to align the reference fingerprint with the tested frames, we exploit the similarity $\mathcal{T}_\text{v}$ defined in Section 3.1.2 by means of the matrix $\mathbf{T}_\text{v}$.

Therefore, we estimate the scaling factor, the rotation angle and the relative shift for every frame in the set such that the PCE is maximized, i.e.,

$$P_f = \max_{s_\text{v} \in \mathcal{S}_\text{v}, \alpha_\text{v} \in \mathcal{A}_\text{v}, \mathbf{c}_\text{v} \in \mathcal{C}_\text{v}} \text{PCE}(\mathbf{W}_f, \mathbf{I}_f \cdot \mathcal{T}_\text{v}(\mathbf{K}_d)), \tag{3.10}$$

where f is the frame index belonging to the set $\mathcal{F}$ of considered query frames, with cardinality F. Note that the search ranges $\mathcal{S}_\text{v}, \mathcal{A}_\text{v}, \mathcal{C}_\text{v}$ can be different from the ones presented in Section 3.2.1, as in this case both video frames and device fingerprint are in video resolution.

In order to attribute or not the video query to the camera, we simply select the highest P_f over all tested frames as

$$P_\text{comp} = \max_{f \in \mathcal{F}} P_f. \tag{3.11}$$

The variable P_comp is named after the chosen testing strategy, which estimates the *complete* set of parameters describing the similarity transformation between frames and reference fingerprint. If P_comp is above a certain threshold, the query is attributed to the camera, otherwise it is considered coming from a different device. This approach empirically proves to be quite accurate, and we show in our experimental analysis that even a reduced number of frames is enough for performing a correct video query matching.

In particular, notice that if $\mathbf{K}_d = \mathbf{K}_\text{v}$, only one geometric transformation is performed on the device fingerprint in order to register it on each query frame. On the contrary, the case $\mathbf{K}_d = \mathbf{K}_\text{iv}$ requires two consecutive geometric transformations: the former warps the device PRNU $\mathbf{K}$ into $\mathbf{K}_\text{iv}$, the latter registers $\mathbf{K}_\text{iv}$ on each query frame. At a first glance it may seem unnecessary to apply these two transformations separately: indeed, one might think of directly using the original device PRNU $\mathbf{K}$ as $\mathbf{K}_d$ for testing a generic video query. In this case, it would be enough to simply register the high resolution $\mathbf{K}$ on the stabilized video content estimating one transformation $\mathcal{T}_f$ per query frame, as defined in (3.5). However, this process will actually result in worse performance if compared to the use of $\mathbf{K}_\text{iv}$ with the proposed strategy. First, as the fingerprint $\mathbf{K}_\text{iv}$ has reduced resolution, it enables to speed up the process because it reasonably requires less memory usage and less computational power. Moreover, $\mathbf{K}_\text{iv}$ results after some cropping operations performed on the scaled PRNU. This step allows to remove from the reference fingerprint the sensor pixels lying outside the area used for video recording (i.e., the blue frame within the dotted line in Figure 3.3), which negatively affect PCE computation.

Despite the use of either $\mathbf{K}_{\mathrm{iv}}$ or $\mathbf{K}_{\mathrm{v}}$ as device fingerprint $\mathbf{K}_d$, the proposed methodology can still be slightly time consuming, as the estimation of the warping parameters through PS requires a fairly high amount of operations. In order to overcome this issue, we propose one possible way out, which can be very efficient whenever there is a consistent amount F of query frames. Indeed, it is likely that not all frames in the set underwent strong rotation or scale transformations due to stabilization. As reported in [30], it is common to exploit simplified motion models including translation only to stabilize some video frames, at the benefit of faster estimation and higher stability. Hence, we can limit our search to the estimation of the relative shift between the query frames and the fingerprint. To test a video query, we select the best PCE obtained over the set as

$$P_{\text{quick}} = \max_{f \in \mathcal{F}} \text{PCE}(\mathbf{W}_f, \mathbf{I}_f \cdot \mathbf{K}_d). \tag{3.12}$$

To attribute the video query to the device under analysis, we threshold P_{quick}.

Concerning both proposed methods, robustness strongly depends on the length of the video query. The larger the frame-set, the higher the probability to find one correlating frame over the whole video. We refer to the approach described in (3.10) and (3.11) as *complete* test strategy, whereas we depict the procedure shown in (3.12) as *quick* test strategy.

3.2.4 How to tackle maximization problems

The maximization problems shown in (3.6) and (3.10) aim at estimating the transformation parameters between the video frames and the reference fingerprint. We propose to split them in two sequential steps:

1. estimating scale and rotation angle;
2. deriving the relative pixel shift.

This is motivated by the natural robustness of PCE to translation, as the peak coordinates resulting from PCE are an estimate of the mutual shift between the correlated terms. For this reason, we first estimate scale and rotation angle which maximize (3.6) and (3.10), then we directly derive the shift from the coordinates of the maximum correlation peak.

Unfortunately, the behaviour of PCE as a function of scale and rotation can be strongly not convex, actually reporting many local maxima, which hinder the use of gradient descent methods for estimating the correct parameters. For instance, Figure 3.5 shows examples of the behaviour of

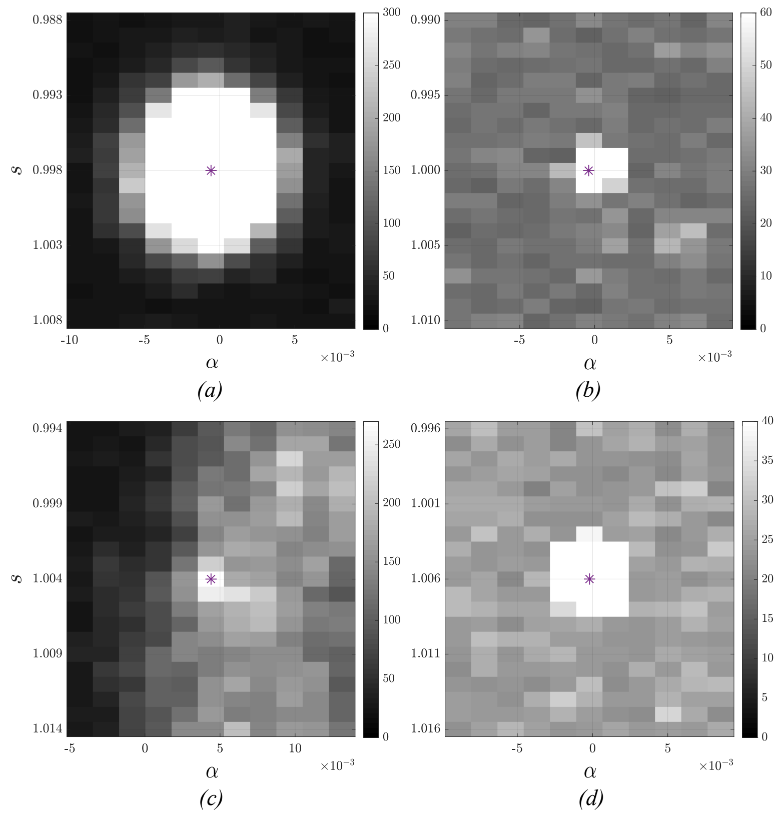


Figure 3.5: *Examples of behaviour of* PCE *for different video frames* (α *is expressed in* [*rad*]). *Specifically, (a) depicts the resulting function selecting* $\mathbf{I}_f$ *from a low-textured and static sequence; in (b),* $\mathbf{I}_f$ *comes from a still video in outdoor scenario; in (c),* $\mathbf{I}_f$ *is taken by a moving user, but in an almost flat scenario; in (d),* $\mathbf{I}_f$ *depicts an outdoor scenario with user motion. Symbol* $*$ *represents the global maximum position.*

PCE according to various values of s and α, computed between stabilized video frames and the fingerprint $\mathbf{K}_{\mathsf{iv}}$ of their original device, warped by the similarity transformation $\mathcal{T}_{\mathsf{v}}(\cdot)$. It is noticeable that, whether $\mathbf{I}_f$ comes from a static and almost flat video, the global maximum is likely to be found, because PCE is basically convex in the space of searched scale and angle. On the contrary, if some texture or user motion are included, PCE presents a global maximum localized within a very small parameter area, and many local maxima can be found, even far from the true one.

In light of these considerations, we propose to search the peak of PCE exploiting global optimization algorithms. In particular, we investigated three different strategies: a particle swarm (PS) [93], a genetic algorithm

(GA) [94] and a multi start (MS) [95] method. We verified that PS is the best solution, in terms of achieved device attribution accuracy and computational burden. For this reason, we exploit PS for all the experiments, even though we dedicate a vast space of comparison between the algorithms for what concerns the problem of attributing a stabilized video to the correct device fingerprint. In the following, all the details about the experimental set-up used for the performances evaluation.

3.2.5 Experiments and results

In this section we report the results of the conducted experimental analysis. First, we describe the used dataset, then we define the adopted evaluation metrics and optimization strategy, finally we report the numerical results achieved by the proposed method. In doing so, we also compare with state-of-the-art methods for device identification using stabilized video sequences.

Dataset

In order to test our method in a fair setup, we make use of a dataset of almost 400 videos coming from 24 different devices. This dataset has been built starting from the recently released Vision Dataset, which includes images and videos from a wide variety of mobile devices from 11 major brands [29]. Specifically, before starting with the investigations, we synchronize the imagery of each device in landscape format.

To build the image PRNU $\mathbf{K}$ for each device, we select all the available images shot by the device depicting scenes of flat surfaces. To be precise, at least 100 images of this nature are available for each device.

Concerning videos, we select all devices whose video resolution is equal to Full-HD (1920×1080 pixel). We consider both static and motion scenes (corresponding to the tags *still, panrot, move* in [29]). Moreover, we also include videos with almost-flat content and with a significant texture contribution (i.e., labeled as *flat, indoor, outdoor* in [29]). Therefore, we end up with 165 non-stabilized sequences from 10 devices, and 232 stabilized video sequences from 14 devices. For each sequence, with an average temporal duration of one minute, we only exploit I-frames, as they contain more reliable sensor noise information with respect to inter-predicted frames [13, 83]. When we refer to any specific device, we use the same naming convention introduced in [29].

Evaluation metrics

In order to assess the accuracy in solving camera attribution problem, we resort to receiver operating characteristic (ROC) curves. Specifically, for each camera, we consider all videos recorded with that camera as positive samples, whereas the set of negatives includes an equal amount of sequences not taken with that camera, randomly selected from the dataset. Each curve depicts the resulting relationship between true positive rate (TPR) and false positive rate (FPR), averaged over the set of available cameras. To numerically evaluate the quality of the attribution, we use:

- AUC, defined as the area under the (ROC) curve.
- $\mathrm{TPR}_{@0.01}$, defined as the TPR calculated at a fixed FPR of 1%.

The goal is to achieve a high value of AUC (ideally 1) and the highest possible value for $\mathrm{TPR}_{@0.01}$ as well.

Particle swarm optimization

Maximization problems are solved using a particle swarm optimizer (PS), which is a population-based algorithm, where a collection of individuals called particles move in steps throughout a search region [96, 97]. At each step, the algorithm evaluates the objective function for every particle, and the best particle corresponding to the maximum of the function is selected. Then, particles move, and after a number of iterations the global maximum is hopefully found. In particular, we tackle maximization problems using a common parameter configuration for the PS, shared by all the following reported experiments. This configuration allows a reliable estimation of the maximum of the considered functions, and consists in:

- $N_p = 50$, i.e., the number of used particles;
- $\max_{it} = 50$, i.e., the maximum number of PS iterations.

The other parameters are the default ones defined in [97].

Preliminary study on stabilization disadvantages

In order to confirm the challenge of dealing with stabilized video sequences, we perform a preliminary test consisting in facing camera attribution problem using standard procedures devised for non-stabilized videos.

Similarly to approaches proposed in [23, 29], we compute the fingerprint of each video by simply aggregating noise residuals extracted from I-frames. The reference camera fingerprint is estimated selecting a static

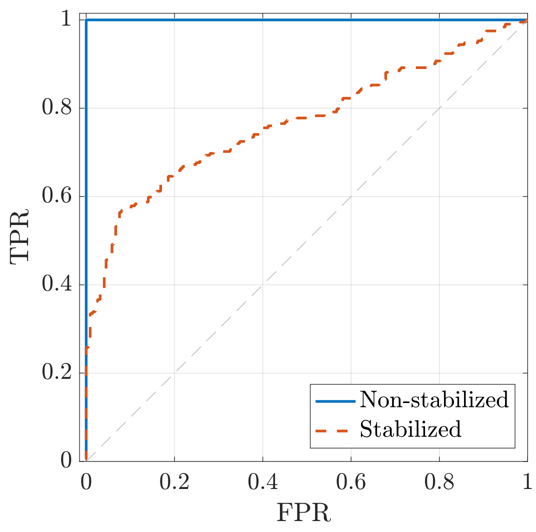


Figure 3.6: *ROC curves obtained using the standard PRNU-based video source attribution method [23] considering non-stabilized and stabilized sequences. Video stabilization strongly hinders the state-of-the-art performance.*

and low-textured sequence for each device, precisely the first video tagged as *flat-still* in the dataset. For testing a generic video query, we compute the PCE between the camera fingerprint and the query one. We apply the same pipeline to both non-stabilized and stabilized video sequences.

Results are reported in Figure 3.6. The difference between stabilized and non-stabilized videos ROC curves is clear. For the non-stabilized pool, the pipeline achieves $\mathrm{AUC} = 1$, which means perfect device attribution. For the stabilized set, this pipeline achieves lower performance with $\mathrm{AUC} = 0.77$. This confirms that video stabilization makes PRNU-based video camera attribution a more challenging task as shown in [12, 13].

Considerations about the first video frame

The authors of [30] show that, generally, the first frame of a video does not undergo any stabilization, as there is no motion to be corrected. Indeed, it is possible that the first frame is taken as reference for stabilizing the next frames. On one hand, this is good news whenever it is available to the analyst. On the other hand, we must realistically assume that the video sequence might have been temporally trimmed, thus making the first acquired video frame unavailable.

In order to test the effect of using or not the first frame for camera attribution with stabilized videos, we perform the following experiment. We

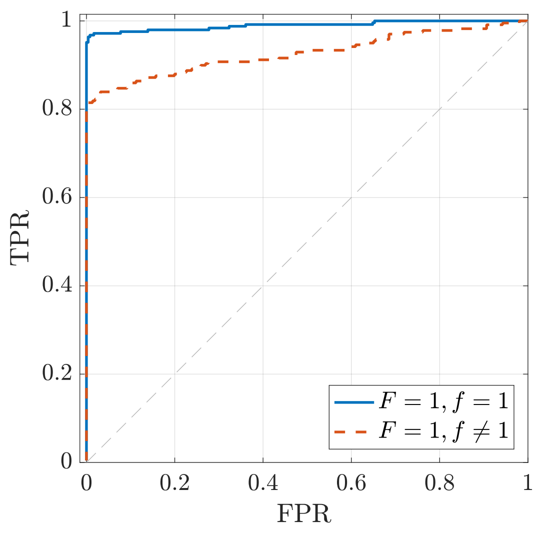


Figure 3.7: *ROC curve related to a single-frame video, exploiting the complete test strategy.*

estimate the reference fingerprint $\mathbf{K}_{\mathrm{iv}}$ for each device.

Then, we proceed with the testing phase, following both the complete and the quick strategy reported in Section 3.2.3, considering a single frame for each video query (i.e., $F = 1$). The test has been performed in two scenarios: (i) selecting the first frame (i.e., $f = 1$); (ii) selecting a random I-frame different from the first one (i..e, $f \neq 1$).

ROC curves for the complete test strategy are reported in Figure 3.7, while ROC curves for the quick test strategy are reported in Figure 3.8. In both situations, results obtained considering the first frame are well above the ones computed with a random frame. This confirms that stabilization algorithms used by devices within Vision dataset may skip stabilization on the first frame.

In order to avoid biasing the results and making wrong conclusions about the proposed algorithm, hereinafter we never include the first frame in our experiments, neither for the estimation of the video camera fingerprint, nor for the attribution phase. In doing so, we assume working in the far more general scenario in which videos available at the analyst can be short portions of longer sequences.

Reference video fingerprint from images and videos

To build the video fingerprint $\mathbf{K}_{\mathrm{iv}}$ in Full-HD resolution, we need to estimate some scale and translation parameters for each device as explained

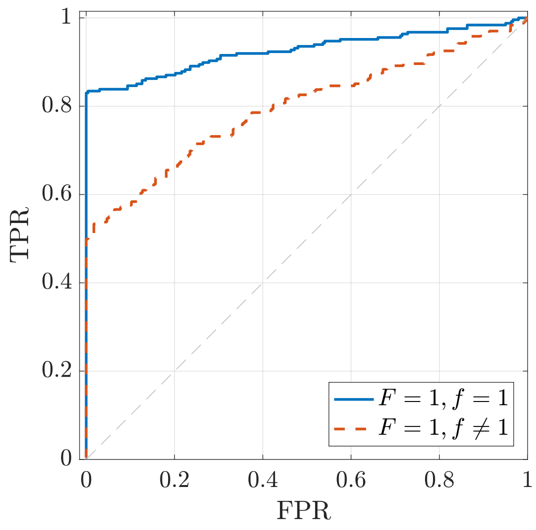


Figure 3.8: *ROC curve related to a single-frame video, exploiting the quick test strategy.*

in Section 3.2.1. To estimate the image to video domain transformation, we apply the proposed $\mathbf{K}_{\text{iv}}$ estimation algorithm to 10 randomly selected I-frames coming from reasonably flat and almost static scenes from each stabilized device in the dataset. Precisely, we pick them from videos tagged as *flat-still* in [29]. Then, we choose as reasonable search range for the scaling factor $\mathcal{S} = [0.3, 0.85]$, whereas the shift is searched all over the video resolution. The boundaries of the search range $\mathcal{S}$ are chosen in order to respect some specific constraints. First, the final resolution after scaling should not be less than Full-HD, otherwise some zero-padding would be required. Secondly, after scaling, the following cropping operation for obtaining the Full-HD resolution should not reasonably discard an excessive percentage of the original sensor area. Therefore an upper bound for the scale must be set. Actually, these bounds vary according to the device image resolution and other device specifications. In this work, the reported bounds are inclusive of all the stabilized devices in the dataset.

Table 3.1 reports the estimated average scale (i.e., s_{iv}) and translation (i.e., c_{iv_x} and c_{iv_y}) parameters for each stabilized device. In particular, we estimate the conversion parameters by running the PS in parallel over multiple cores. In our workstation composed by $2\times$Intel Xeon Gold 6126 2.6GHz with 377GB of RAM running Ubuntu 17.10, the process takes on average $47.4\,$s per frame. Notice that, for each device, we exploit images with original resolution reported in [29]. Actually, since a lot of devices allow to record photos and videos with various resolutions, the conversion

Table 3.1: *Average scaling and translation parameters for image-to-Full-HD-video domain conversion. Device naming convention is the same as in [29]. Only stabilized devices have been used.*

DEVICE	D02	D05	D06	D10	D12	D14	D15	D18	D19	D20	D25	D29	D32	D34
s_{iv}	0.75	0.687	0.707	0.75	0.379	0.688	0.706	0.688	0.706	0.815	0.517	0.687	0.517	0.687
c_{iv_x}	270	158	201	279	33	167	190	166	190	97	239	161	242	161
c_{iv_y}	374	304	328	384	205	308	323	308	324	248	361	302	356	302

parameters for each device are not fixed, but depend on the image-video resolution under investigation. Therefore, the transformation $\mathcal{T}_{\mathsf{iv}}$ is generally not unique and should be computed every time either image or video resolution changes.

Reference video fingerprint from videos only

In order to estimate the camera fingerprint $\mathbf{K}_{\mathsf{v}}$ from stabilized videos only, we follow the iterative noise aggregation method proposed in Section 3.2.2. In particular, for each stabilized device, we select an almost static video with little image content (precisely, the first video flagged as *flat-still* in the dataset), considering different values for the parameter Δ limiting the shift search range.

We expect $\mathbf{K}_{\mathsf{v}}$ to become a better and better estimate of the true device fingerprint as long as we correctly aggregate more frames. To evaluate whether the aggregation method is properly working, we define $\rho(f)$ as the normalized cross-correlation (NCC) between the fingerprint $\mathbf{K}_{\mathsf{iv}}$ and the fingerprint $\mathbf{K}_{\mathsf{v}}(f)$, estimated using f frames. Notice that we should not directly cross-correlate these terms, as $\mathbf{K}_{\mathsf{v}}$ has been built by selecting a certain frame as reference, and then registering other frames on it. Therefore, because of video stabilization, the obtained fingerprint $\mathbf{K}_{\mathsf{v}}$ could be slightly scaled, rotated and shifted with respect to the fingerprint $\mathbf{K}_{\mathsf{iv}}$. In order to compare these terms, we estimate the similarity transformation which registers the fingerprint $\mathbf{K}_{\mathsf{v}}$ on $\mathbf{K}_{\mathsf{iv}}$, then we apply this transformation to the frame-variant $\mathbf{K}_{\mathsf{v}}(f)$.

Actually, $\rho(f)$ can be very helpful to evaluate the algorithm performance in estimating the video fingerprint. For instance, if $\rho(f)$ has a monotonic increasing behavior, we are aggregating the frames in a correct way. Otherwise, we are aligning frames by means of some correlating content not due to the original device fingerprint. The choice of NCC as metrics instead of the PCE is motivated by its higher computational efficiency. Moreover, being normalized at 1, NCC allows a clearer comparison between the per-

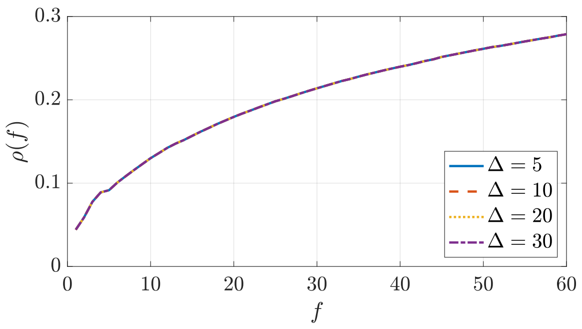


Figure 3.9: *Resulting cumulative* NCC *computed between* $\mathbf{K}_{\mathsf{iv}}$ *and registered* $\mathbf{K}_{\mathsf{v}}(f)$ *for device D34.*

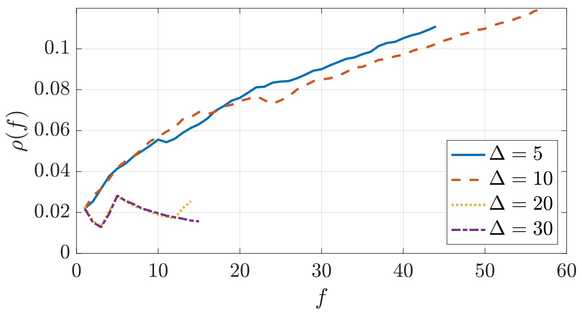


Figure 3.10: *Resulting cumulative* NCC *computed between* $\mathbf{K}_{\mathsf{iv}}$ *and registered* $\mathbf{K}_{\mathsf{v}}(f)$ *for device D32.*

formance of different devices.

We compute $\rho(f)$ for each stabilized device in the dataset. For instance, results for devices D32 and D34 are reported in Figures 3.9 and 3.10 as representative of the overall trend on all videos. For some video sequences (see Figure 3.9) the bound on Δ value does not impact on the algorithm, as all the available I-frames can be correctly aggregated independently from the chosen Δ. However, in case videos contain more texture (see Figure 3.10), the proposed method tends to register scene content rather than sensor noise traces if Δ is too high. Indeed, $\rho(f)$ does not increase. Moreover, it can happen that the aggregation process stops after few frames, as there is only a restricted amount of imagery satisfying the constraints reported in Section 3.2.2*(b)*. For this reason, hereinafter we limit our further analysis to $\Delta = \{5, 10\}$, since these values enable achieving the best aggregation performances on average, i.e., a high final value of $\rho(f)$ and more aggregated frames.

Finally, Figure 3.11 shows the values achieved by $\rho(f)$ at the last ag-

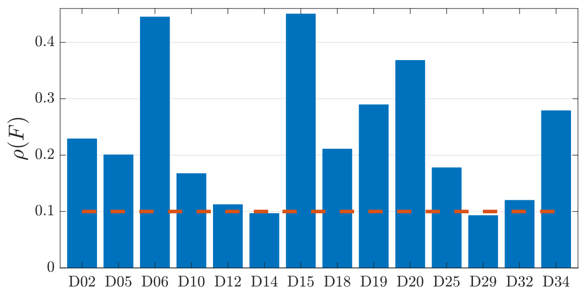


Figure 3.11: *Final value of the resulting cumulative* NCC *for all stabilized devices in the dataset, setting* $\Delta = 10$*.*

gregated frame, namely $\rho(F)$, obtained by fully running the proposed $\mathbf{K}_\mathsf{v}$ estimation method setting $\Delta = 10$ for all devices. Notice that $\rho(F)$ values are almost always higher than 0.1, which actually represents a good NCC measure in standard attribution problems [20]. This confirms that the estimated video fingerprint $\mathbf{K}_\mathsf{v}$ is informative of the camera model.

Testing the video query

To check how accurately we can attribute a stabilized video to the originating device, we present the ROC curves computed over all the stabilized devices in the dataset. To be precise, for each stabilized video sequence, we randomly select F I-frames, and we test the complete and quick attribution methods on both fingerprints $\mathbf{K}_\mathsf{iv}$ and $\mathbf{K}_\mathsf{v}$.

First, we report the results achieved using the complete method reported in Section 3.2.3. Specifically, considering the fingerprint $\mathbf{K}_\mathsf{iv}$, we investigate the previously reported global optimization strategies using a standard set of parameters as defined in [97]. Notice that, to perform a fair comparison between the solvers, we select an equal number of swarm particles for PS, of population size for GA and problem instances for MS, precisely set to 20 (i.e., the default *swarm size* of PS for searching 2 parameters). Then, as all the methods search for the maximum of PCE inside some predefined boundaries in the parameter space, we fix for the three of them a scale range $\mathcal{S}_\mathsf{v} = [0.99, 1.01]$ and a rotation range $\mathcal{A}_\mathsf{v} = [-0.15, 0.15]$rad, following an approach similar to [30]. Indeed, reference fingerprint is already in the video domain, thus we only need to slightly warp frames.

In this case, our experimental set-up includes the stabilized dataset, extracting 1 random I-frame from each video. Figure 3.12(a) shows ROC curves achieved by the three proposed algorithms. Notice that PS exceeds the other two strategies for both AUC and $\mathrm{TPR}_{@0.01}$. This accuracy gain

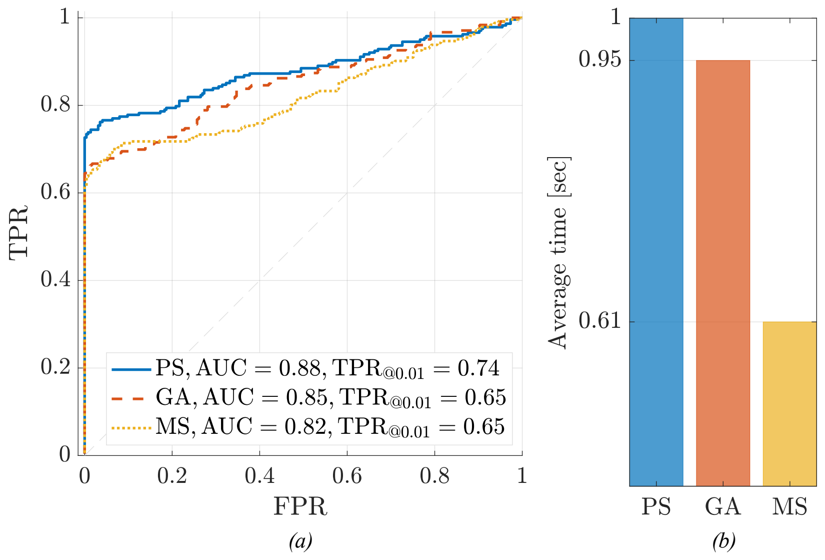


Figure 3.12: *ROC curves (a) and average computation times (b) achieved by PS, GA and MS methods using their default parameters.*

brings a few drawbacks regarding the average computational time, shown in Figure 3.12(b), which is higher for PS (whose time is taken as reference), but still affordable. Hence, we select PS as the primary method for solving the source device identification problem.

Figure 3.13 depicts the results of complete test strategy evaluated using both $\mathbf{K}_{\mathsf{iv}}$ and $\mathbf{K}_{\mathsf{v}}$ as reference fingerprints, and testing $F = \{5, 10\}$ random I-frames of the query videos. Notice that we only show results for $\mathbf{K}_{\mathsf{v}_{\Delta=10}}$ ($\mathbf{K}_{\mathsf{v}}$ computed with $\Delta = 10$) as these are highly comparable to the case $\Delta = 5$. It is possible to note that the proposed method is quite accurate. As a matter of fact, testing just 5 I-frames (i.e., ~ 5 seconds of video content), we obtain $\mathrm{AUC} = 0.96$ exploiting the fingerprint $\mathbf{K}_{\mathsf{iv}}$. The performances achieved by $\mathbf{K}_{\mathsf{v}}$ are quite good as well, considering this fingerprint is computed from video frames only. Nonetheless, the larger the amount of investigated I-frames, the better the ROC curve. For instance, regarding $\mathbf{K}_{\mathsf{v}_{\Delta=10}}$ results, with just 5 frames we achieve $\mathrm{AUC} = 0.89$, whereas 10 frames return $\mathrm{AUC} = 0.92$.

The overall results of the complete strategy are depicted in Table 3.2, which reports the achieved AUC and $\mathrm{TPR}_{@0.01}$ corresponding to all the curves. In particular, on our workstation, the average time for testing 1 query frame against 1 camera fingerprint with the complete strategy (considering matching cases as well as non-matching ones) is of $57\,\mathrm{s}$, therefore

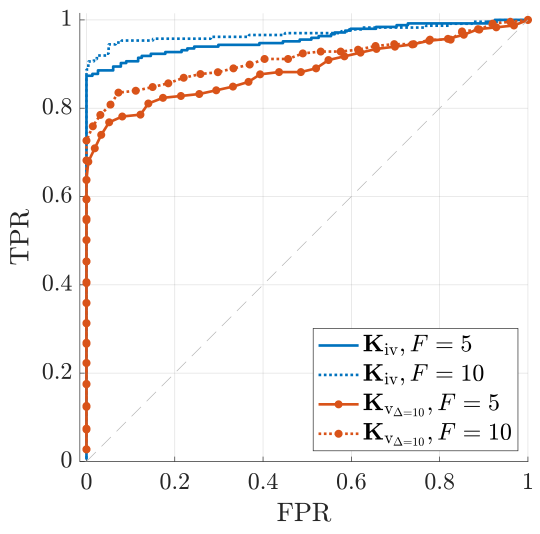


Figure 3.13: *ROC curves obtained testing $F = \{5, 10\}$ I-frames with the complete strategy.*

Table 3.2: AUC *and* $\text{TPR}_{@0.01}$ *exploiting* $\mathbf{K}_{\text{iv}}$ *and* $\mathbf{K}_{\text{v}}$ *as reference fingerprint, testing* $F = \{5, 10\}$ *random I-frames with the complete strategy.*

FINGERPRINT	$\mathbf{K}_{\text{iv}}, F = 5$	$\mathbf{K}_{\text{iv}}, F = 10$	$\mathbf{K}_{\text{v}_{\Delta=5}}, F = 5$	$\mathbf{K}_{\text{v}_{\Delta=5}}, F = 10$	$\mathbf{K}_{\text{v}_{\Delta=10}}, F = 5$	$\mathbf{K}_{\text{v}_{\Delta=10}}, F = 10$
AUC	0.96	0.97	0.9	0.89	0.89	0.92
$\text{TPR}_{@0.01}$	0.87	0.91	0.71	0.77	0.7	0.73

the process requires on average less than 5 minutes for testing 5 frames. To be more specific, the computational time derives from the computation of one geometric transformation (i.e., $\mathcal{T}_{\text{v}}$) and one PCE evaluation for each particle at each iteration of the PS algorithm. In addition, we show the results obtained with the alternative quick attribution method proposed in Section 3.2.3. Figure 3.14 depicts the results evaluated using both reference fingerprints. Specifically, we select $F = \{5, 20, 50\}$ query I-frames for evaluating the performance of the fingerprint $\mathbf{K}_{\text{iv}}$, whereas for the fingerprint $\mathbf{K}_{\text{v}}$ we limit the plot to the use of 50 I-frames, as a lower amount of frames reduces the performance. Note that some sequences in the dataset do not have 50 I-frames. In these situations we use as many I-frames as possible. We can notice that, to obtain results with similar accuracy to those of the complete testing procedure, 50 I-frames are needed rather than just 5. Anyway, even exploiting $\mathbf{K}_{\text{v}}$ as reference, we can solve the attribution problem using the quick algorithm, as long as the analyst has approximately one minute long video.

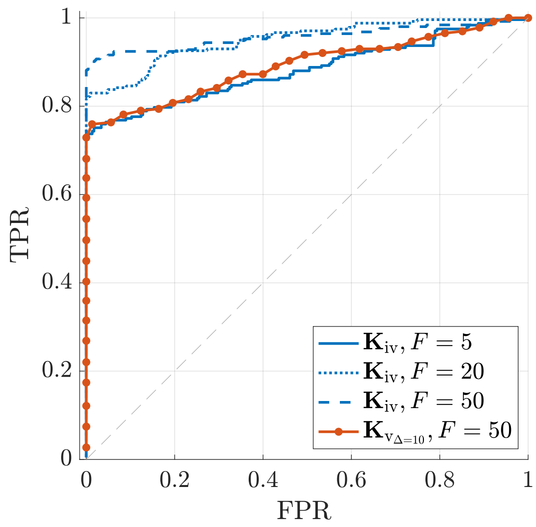


Figure 3.14: *ROC curves obtained testing $F = \{5, 20, 50\}$ I-frames with the quick strategy.*

Table 3.3: AUC *and* $\mathrm{TPR}_{@0.01}$ *exploiting* $\mathbf{K}_{\mathrm{iv}}$ *and* $\mathbf{K}_{\mathrm{v}}$ *as reference fingerprint, testing* 50 *random I-frames following the quick test strategy.*

FINGERPRINT	$\mathbf{K}_{\mathrm{iv}}$	$\mathbf{K}_{\mathrm{v}_{\Delta=5}}$	$\mathbf{K}_{\mathrm{v}_{\Delta=10}}$
AUC	0.96	0.88	0.89
$\mathrm{TPR}_{@0.01}$	0.89	0.74	0.75

To be precise, Table 3.3 contains AUC and $\mathrm{TPR}_{@0.01}$ corresponding to the curves achieved testing 50 query I-frames with both reference fingerprints $\mathbf{K}_{\mathrm{iv}}$ and $\mathbf{K}_{\mathrm{v}}$. Concerning the computational time, the quick strategy requires only one PCE evaluation per query frame, ending up with an average time for testing 1 frame against 1 camera fingerprint of $0.075\,\mathrm{s}$. Thus, for 50 query frames we need less than $4\,\mathrm{s}$.

As far as the comparison between the complete and quick methods is concerned, we can notice that the former method is more accurate than the latter one as expected. Indeed, in order to fairly compare the two methods, we should consider the very same number of testing frames for the both of them. As a matter of fact, exploiting just 5 I-frames returns similar results to the case in absence of rotation and scaling only for $F = 50$. For this reason, when few video frames are available, we suggest to estimate the similarity transformation between the camera signature and the frames. Indeed, accuracy is increased at the expense of more computational time.

On the contrary, when plenty of frames are at hand, it can be a good choice to limit the analysis to translation models, since the pair $\mathrm{AUC} - \mathrm{TPR}_{@0.01}$ reports highly acceptable and comparable results with the first solution.

State-of-the-art Comparison

To the best of our knowledge, only few methods have been proposed in the literature to deal with camera attribution from stabilized videos.

One solution has been presented in [13]. The authors consider videos stabilized by means of a controlled algorithm (i.e., FFMPEG deshaker), which only applies rotations and translations. As the proposed method does not take scaling into account and does not deal with videos directly stabilized on the recording device, it is likely going to fail on the uncontrolled dataset used in this thesis.

A more recent solution has been proposed in [12]. The authors propose to search for scales and translations, but they do not take rotations into account. Moreover, they only attribute stabilized videos to cameras if a reference PRNU obtained from still images is available (i.e., they do not compare videos to videos). This makes their problem formulation more similar to the one we described in Section 3.2.1, rather than the method proposed in Section 3.2.2.

Additionally, both solutions presented in [12, 13] make use of the first frame of each video sequence, which we do not consider as there is a high chance it has not been stabilized, thus making the problem less challenging.

In the light of these considerations, even the comparison against [12] would not be completely fair. However, the used metrics are the same (i.e., $\mathrm{TPR}_{@0.01}$ and AUC), and concerning the dataset, we both consider videos from the Vision dataset (8 devices in [12], 14 devices in this work). Therefore, a few conclusions can still be drawn. To compare the methods in the same experimental setup, we select from Vision dataset all the available instances we can find for each device model exploited in [12]. Exploiting the video fingerprint $\mathbf{K}_{\mathsf{iv}}$ described in Section 3.2.1 over this reduced dataset, we are able to achieve $\mathrm{TPR}_{@0.01} = 0.89$ and $\mathrm{AUC} = 0.96$ using the complete strategy on 5 query frames, and $\mathrm{TPR}_{@0.01} = 0.92$ and $\mathrm{AUC} = 0.97$ testing 50 query frames with the quick method. Conversely, results in [12] show $\mathrm{TPR}_{@0.01} = 0.87$ and $\mathrm{AUC} = 0.95$, which are below the ones achieved by us, even considering that we are discarding the contribution from the first frame, and we also cope with the video vs. video case.

Upper bound on video fingerprint estimation

In order to understand whether it is possible to extract better fingerprint information from stabilized video frames, we perform a final experiment involving an *Oracle* providing us with data normally unavailable to the analyst.

Specifically, let us consider the scenario depicted in Section 3.2.2, in which the video fingerprint $\mathbf{K}_{\text{v}}$ is extracted from video frames. However, we envision an *Oracle* telling us how to align each frame noise residual with the others, in order to obtain a much better video fingerprint estimate. To do this practically, we apply an analogous algorithm to the one proposed in Section 3.2.2, with the difference that frame alignment step is performed by similarity transformation against the reference $\mathbf{K}_{\text{iv}}$ (i.e., a cleaner version of the device fingerprint) rather than a reference video frame.

Of course, this is clearly an unrealistic situation (i.e., if the analyst had $\mathbf{K}_{\text{iv}}$, he/she could use the algorithm proposed in Section 3.2.1). Nonetheless, this is a very powerful investigation tool for evaluating the accuracy of our results. We can therefore compare results obtained with this *Oracle*-based fingerprint, and with the proposed fingerprint $\mathbf{K}_{\text{v}}$, to see how much they differ.

In terms of quality of the estimated fingerprint, the final values of $\rho(F)$ evaluated with the *Oracle*-based fingerprint only report a slight increment (less than 0.1 on average) with respect to those obtained in Figure 3.11 using $\mathbf{K}_{\text{v}}$. This confirms that the proposed method is quite good and represents a viable solution for extracting the device fingerprint in video domain.

In terms of device attribution, Figure 3.15 reports ROC results obtained with the use of the *Oracle* and results obtained with $\mathbf{K}_{\text{v}}$, using either the complete or quick test methods. The $\text{TPR}_{@0.01}$ of *Oracle*-based strategy reaches 0.84, while the $\text{TPR}_{@0.01}$ of complete strategy using $\mathbf{K}_{\text{v}}$ is 0.14 points lower, but the quick strategy is able to achieve $\text{TPR}_{@0.01} = 0.76$. Notice that the accuracy gap between complete and quick strategies should not lead to rushed conclusions: as previously stated, the two methods can be correctly compared only if the number of query frames is the same. Considering the overall performance, the proposed method can be considered reasonably accurate, taking into account that it works in the realistic scenario where video sequences can contain some textures, potentially undermining the video fingerprint estimation.

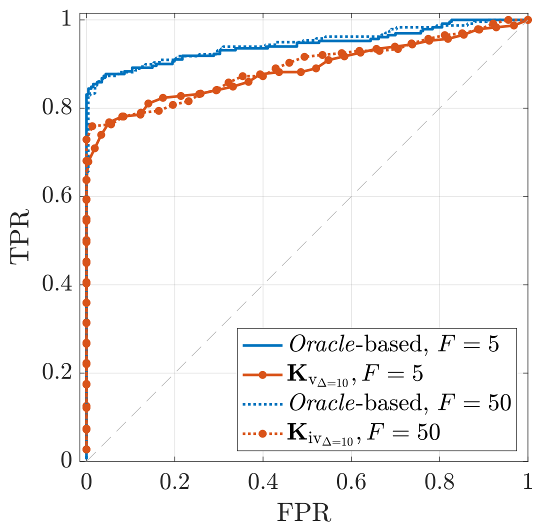


Figure 3.15: *ROC curves obtained testing the complete strategy on* 5 *I-frames, and testing the quick strategy on* 50 *I-frames, using the* Oracle-*based fingerprint and* $\mathbf{K}_\mathsf{v}$.

3.2.6 A Fourier-Mellin approach to source device identification

In this section, we present an alternative solution for solving the complete test strategy for source device identification. Specifically, we leverage the so-called Fourier-Mellin transform [98] to solve the maximization problem shown in (3.10). In the following, we present the theoretical motivations behind this choice, then we describe our approach and report a comparison with the previously shown strategy.

Fourier-Mellin transform

Fourier-Mellin transform exploits the properties of Fourier transform with respect to scale, rotation and shift for estimating the similarity transformation between two images [98]. If an image $\mathbf{I}_\mathsf{a}$ is denoted as $\mathcal{T}_\mathsf{ab}(\mathbf{I}_\mathsf{b})$, where $\mathcal{T}_\mathsf{ab}$ is a similarity transformation including a scaling factor s_ab, a rotation angle α_ab and shift vector $\mathbf{c}_\mathsf{ab}$, it is theoretically possible to estimate this transformation in closed form, given $\mathbf{I}_\mathsf{a}$ and $\mathbf{I}_\mathsf{b}$ only. For the sake of clarity, we denote with $\mathbf{T}_\mathsf{ab}$ the matrix corresponding to similarity $\mathcal{T}_\mathsf{ab}$:

$$\mathbf{T}_\mathsf{ab} = \begin{pmatrix} s_\mathsf{ab} \cdot \cos\alpha_\mathsf{ab} & -s_\mathsf{ab} \cdot \sin\alpha_\mathsf{ab} & c_{\mathsf{ab}_x} \\ s_\mathsf{ab} \cdot \sin\alpha_\mathsf{ab} & s_\mathsf{ab} \cdot \cos\alpha_\mathsf{ab} & c_{\mathsf{ab}_y} \end{pmatrix}. \tag{3.13}$$

Notice that $\mathbf{T}_\mathsf{ab}$ models as initial geometric transformations the scale and rotation, only later including translation as successive transformation.

In absence of rotation and scaling, translation can be estimated using the Fourier shift theorem [98]. Whether $\mathbf{I}_\mathsf{a}$ has been translated with respect to $\mathbf{I}_\mathsf{b}$ of $\mathbf{c}_\mathsf{ab} = (c_{\mathsf{ab}_x}, c_{\mathsf{ab}_y})$, Fourier transform fulfills this condition:

$$\mathbf{F}_\mathsf{a}(f_x, f_y) = \mathbf{F}_\mathsf{b}(f_x, f_y) \cdot e^{-j2\pi c_{\mathsf{ab}_x} f_x - j2\pi c_{\mathsf{ab}_y} f_y}, \tag{3.14}$$

being $\mathbf{F}_\mathsf{a}$ and $\mathbf{F}_\mathsf{b}$ the Fourier transforms of $\mathbf{I}_\mathsf{a}$ and $\mathbf{I}_\mathsf{b}$ and $f_x, f_y \in [-0.5, 0.5]$ the spatial frequencies. The Fourier transform magnitudes of the two images are equal. The only difference consists of a linear phase term as a function of spatial frequencies, due to the mutual shift. This shift can be estimated exploiting phase correlation, defined as the inverse Fourier transform of the cross-power spectrum of the images. More formally, the cross-power spectrum $\mathbf{X}$ is defined as

$$\mathbf{X}[\mathbf{F}_\mathsf{a}, \mathbf{F}_\mathsf{b}] = \frac{\mathbf{F}_\mathsf{a}\mathbf{F}_\mathsf{b}^*}{|\mathbf{F}_\mathsf{a}\mathbf{F}_\mathsf{b}^*|}, \tag{3.15}$$

where the symbol * denotes the complex conjugate and division is computed element-wise. For the sake of notational compactness, we omit the indexes (f_x, f_y) except for cases in which they are useful to clarify equations. Phase correlation $\mathbf{\Phi}$ is the inverse Fourier transform of $\mathbf{X}$, hence:

$$\mathbf{\Phi}[\mathbf{F}_\mathsf{a}, \mathbf{F}_\mathsf{b}] = F^{-1}\{\mathbf{X}[\mathbf{F}_\mathsf{a}, \mathbf{F}_\mathsf{b}])\}, \tag{3.16}$$

being $F^{-1}\{\cdot\}$ the operator of the inverse Fourier transform. In this specific case,

$$\mathbf{X}[\mathbf{F}_\mathsf{a}, \mathbf{F}_\mathsf{b}](f_x, f_y) = e^{j2\pi c_{\mathsf{ab}_x} f_x + j2\pi c_{\mathsf{ab}_y} f_y}, \tag{3.17}$$

thus the phase correlation coincides with an impulse centered in $\mathbf{c}_\mathsf{ab} = (c_{\mathsf{ab}_x}, c_{\mathsf{ab}_y})$. The relative pixel shift $\mathbf{c}_\mathsf{ab}$ can be found as the position of this impulse.

The Fourier-Mellin method also allows to match images which have been rotated one with respect to the other. For instance, if $\mathbf{I}_\mathsf{a}$ is rotated by α_ab with respect to $\mathbf{I}_\mathsf{b}$, the relationship between their transforms becomes

$$\mathbf{F}_\mathsf{a}(f_x, f_y) = \mathbf{F}_\mathsf{b}(f_x \cdot \cos\alpha_\mathsf{ab} + f_y \cdot \sin\alpha_\mathsf{ab}, -f_x \cdot \sin\alpha_\mathsf{ab} + f_y \cdot \cos\alpha_\mathsf{ab}). \tag{3.18}$$

A rotation by an angle α_ab in pixel domain implies that Fourier transform is rotated as well by the same angle. In order to estimate the angle, Fourier-Mellin approach converts this relationship in polar coordinates:

$$P\{\mathbf{F}_\mathsf{a}\}(\rho, \alpha) = P\{\mathbf{F}_\mathsf{b}\}(\rho, \alpha - \alpha_\mathsf{ab}), \tag{3.19}$$

being $P\{\cdot\}$ the operator computing the polar coordinate transformation, ρ the radial coordinate and α the rotational coordinate. It is worth noting that rotation in Cartesian domain becomes translation in polar domain. To estimate α_{ab}, we resort again to phase correlation computed between the Fourier transforms written in polar domain. The y-coordinate of the impulse corresponds to the rotation angle.

A scale transformation by a factor s_{ab} can also be modeled. In this case, according to the Fourier scaling theorem,

$$\mathbf{F}_{\mathsf{a}}(f_x, f_y) = \frac{1}{s_{\mathsf{ab}}} \mathbf{F}_{\mathsf{b}}(f_x/s_{\mathsf{ab}}, f_y/s_{\mathsf{ab}}). \tag{3.20}$$

Given an image stretch in pixel domain, the Fourier transform is squeezed in frequency domain by the same factor. This very same relationship can be written differently, resorting to log-polar coordinates:

$$LP\{\mathbf{F}_{\mathsf{a}}\}(\rho, \alpha) = LP\{\mathbf{F}_{\mathsf{b}}\}(\rho - \log s_{\mathsf{ab}}, \alpha), \tag{3.21}$$

where $LP\{\cdot\}$ is the operator for log-polar transformation. To find s_{ab} it is enough to evaluate again the impulse position in the phase correlation matrix between $LP\{\mathbf{F}_{\mathsf{a}}\}$ and $LP\{\mathbf{F}_{\mathsf{b}}\}$, and convert the found value from logarithmic to linear domain.

Merging all these considerations, Fourier-Mellin strategy allows to estimate the complete similarity transformation (including scale, rotation and translation) between images $\mathbf{I}_{\mathsf{a}}$ and $\mathbf{I}_{\mathsf{b}}$. The workflow consists of two consequent steps:

1. estimate the scale and rotation;
2. estimate the mutual shift.

Initially, selecting only the Fourier magnitudes, we come out with

$$LP\{\mathbf{M}_{\mathsf{a}}\}(\rho, \alpha) = LP\{\mathbf{M}_{\mathsf{b}}\}(\rho - \log s_{\mathsf{ab}}, \alpha - \alpha_{\mathsf{ab}}), \tag{3.22}$$

where $\mathbf{M}_{\mathsf{a}}$ and $\mathbf{M}_{\mathsf{b}}$ represent the Fourier transform magnitudes. Since in log-polar coordinates scale and rotation become shift terms, these can be estimated by looking at the position of the maximum peak of phase correlation computed between the Fourier magnitudes. Images can be realigned for what concern rotation and scale, leaving translation only to be estimated.

Then, as shown in (3.17), the relative shift can be found by examining the coordinates of the maximum peak of phase correlation between the two realigned images.

Going back to source device identification, we can think at the Fourier-Mellin strategy has a potential competitors to the particle swarm approach proposed in Section 3.2.4. Indeed, since the goal of complete test strategy is estimating the geometric transformation that realigns the query frame noise residuals and the device reference fingerprint, we may exploit Fourier-Mellin strategy to derive this transformation following the above steps. For the sake of clarity, in the following section we provide more details about the proposed method.

Application of Fourier-Mellin to the complete test strategy

In this section, we investigate the performances of Fourier-Mellin approach on the source device identification problem. In particular, we apply this strategy to the complete test, exploiting the best device reference fingerprint, i.e., $\mathbf{K}_d = \mathbf{K}_{\text{iv}}$.

At first glance, the solution to this problem may seem straightforward. However, notice that, differently from the Fourier-Mellin theory presented above, the two terms to compare are not exactly one the transformed version (by means of a similarity transformation) of the other. First, as stated in Section 3.1.1, the geometric transformation introduced by stabilization on video frames is not necessarily a similarity, but can include also perspective distortions. Second, the noise residuals of video frames may contain scene content and noise contributions which are not present in the reference device fingerprint. This is typically due to imperfections in the noise extraction process. Indeed, $\mathbf{W}_f$ is usually slightly influenced by the scene depicted on the corresponding frame $\mathbf{I}_f$ and by compression artifacts.

The primary consequence of this dissimilarity is that selecting only the Fourier magnitudes for estimating scale and rotation between the two terms, as reported in (3.22), may be not precise. Indeed, the phase correlation between the Fourier magnitudes of $\mathbf{K}_{\text{iv}}$ and $\mathbf{W}_f$ does not show a pronounced peak, thus the estimation of scale and rotation angle is strongly hindered. A wrong estimation of scale and rotation inevitably leads to a totally wrong estimation of the mutual shift and to a poor performance of the complete test strategy related to the query frame.

In order to overcome this issue, we propose to consider phase terms in addition to Fourier transform magnitude.

Unfortunately, the natural drawback of this approach is that we cannot isolate anymore the estimation of scale and rotation to that of mutual shift. Indeed, in this case, the complete phase correlation does not exclusively depends on scale and rotation transformations, but also on translation between the two terms. Thus, we cannot simply select the two entire Fourier spectra (i.e., magnitude and phase), convert into log-polar domain, compute the phase correlation, select the peak coordinates and find the scale and rotation parameters. This pipeline would work only if $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ were almost perfectly aligned in terms of translation, i.e., if their mutual shift *after correcting scale and rotation* were basically 0 pixels in both horizontal and vertical directions. In other words, we first would have to correctly realign the PRNU traces left on the noise residual with those on the reference fingerprint for what concerns the relative shift, then we could convert the Fourier transforms into log-polar domain and estimate the remaining parameters.

Actually, finding a rule to estimate the relative shift between $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ *without knowing the correct scale and rotation* is far from being an easy task. Indeed, as stated before, the estimation problem is no more separable and shift can be derived only if rotation and scale are estimated too. In order to estimate the best shift parameter, we propose to set a double maximization problem. Formally,

$$c_{\mathsf{fm}_x}, c_{\mathsf{fm}_y} = \arg\max_{c_x, c_y \in \mathcal{C}_{\mathsf{fm}}} (\\ \max_{s,\alpha} \mathbf{\Phi}[LP_{s,\alpha}\{\mathbf{F}_{\mathbf{W}_f}\}, LP_{s,\alpha}\{\mathbf{F}_{\mathbf{K}_{\mathsf{iv}}}(f_x, f_y) \cdot e^{-j2\pi c_x f_x - j2\pi c_y f_y}\}]) \quad (3.23)$$

being $\mathbf{F}_{\mathbf{K}_{\mathsf{iv}}}$ and $\mathbf{F}_{\mathbf{W}_f}$ the Fourier transforms (complete of magnitude and phase) of $\mathbf{K}_{\mathsf{iv}}$ and $\mathbf{W}_f$, respectively. $LP_{s,\alpha}\{\cdot\}$ represents the log-polar transformation as a function of radial and angular coordinates. To clarify, (3.23) follows these steps:

a) fix one shift combination (c_x, c_y), considering $c_x, c_y \in \mathcal{C}_{\mathsf{fm}}$;

b) realign $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ for the selected pixel shift (c_x, c_y), multiplying $\mathbf{F}_{\mathbf{K}_{\mathsf{iv}}}(f_x, f_y)$ by $e^{-j2\pi c_x f_x - j2\pi c_y f_y}$;

c) convert the Fourier transforms into log-polar domain;

d) evaluate phase correlation and select the highest peak.

Repeat steps from *(a)* to *(d)* for each possible shift combination.

e) select the shift combination which returns the highest peak in phase correlation.

Each shift combination is associated to one peak value in phase correlation and a pair of estimated scale and rotation parameters. For the sake of clarity, we denote with $\mathbf{\Phi}_{peak}$ the value of the peak in phase correlation as a function of the mutual shift:

$$\mathbf{\Phi}_{peak}(c_x, c_y) = \max_{s,\alpha} \mathbf{\Phi}[LP_{s,\alpha}\{\mathbf{F}_{\mathbf{W}_f}\}, LP_{s,\alpha}\{\mathbf{F}_{\mathbf{K}_{\mathsf{iv}}} \cdot e^{-j2\pi c_x f_x - j2\pi c_y f_y}\}]. \quad (3.24)$$

The best mutual shift can be found in correspondence of the maximum between all the analyzed peak values, therefore (3.23) becomes

$$c_{\mathsf{fm}_x}, c_{\mathsf{fm}_y} = \underset{c_x, c_y \in \mathcal{C}_{\mathsf{fm}}}{\arg\max} \mathbf{\Phi}_{peak}(c_x, c_y). \tag{3.25}$$

After the shift parameters $(c_{\mathsf{fm}_x}, c_{\mathsf{fm}_y})$ have been estimated, we can search for the scale and rotation parameters. These can be estimated in closed-form by other 3 steps:

f) realign $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ for the found $(c_{\mathsf{fm}_x}, c_{\mathsf{fm}_y})$;

g) convert their Fourier transforms into log-polar domain;

h) look at the coordinates of the maximum peak of their phase correlation.

Formally,

$$\begin{aligned} s_{\mathsf{fm}}, \alpha_{\mathsf{fm}} = \underset{s,\alpha}{\arg\max}(\\ \mathbf{\Phi}[LP_{s,\alpha}\{\mathbf{F}_{\mathbf{W}_f}\}, LP_{s,\alpha}\{\mathbf{F}_{\mathbf{K}_{\mathsf{iv}}}(f_x, f_y) \cdot e^{-j2\pi c_{\mathsf{fm}_x} f_x - j2\pi c_{\mathsf{fm}_y} f_y}\}]). \end{aligned} \tag{3.26}$$

Notice that s_{fm} and α_{fm} can be derived without the need of a global optimization strategy. On the contrary, gradient descent strategies to solve (3.25) suffer from the non-convex behaviour of $\mathbf{\Phi}_{peak}$ as a function of the shift. For instance, Figure 3.16 reports the behaviour of $\mathbf{\Phi}_{peak}$ versus horizontal and vertical shift between some video frames and $\mathbf{K}_{\mathsf{iv}}$. Notice that, especially in video sequences characterized by outdoor scenarios or user motion, the actual maximum peak value can be hard to find with gradient descent algorithms. Many local maxima are spread over all the investigated mutual shift values.

The estimation problem shown in (3.25) could be solved again by resorting to global optimization techniques. It is worth noting that the translation between $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ can be assumed with slight approximation to imply integer shift in horizontal and vertical directions, i.e., to represent a certain number of pixels and not a fraction of them. Particle swarm optimization is very suitable for deriving continuous parameters, however being suboptimal in case of integer estimation. We propose to leave particle swarm in favour of a genetic algorithm that allows a more efficient estimation of integer parameters [97].

Precisely, genetic algorithms mimic biological evolution to solve the selected problem. At each iteration, the algorithm modifies its population:

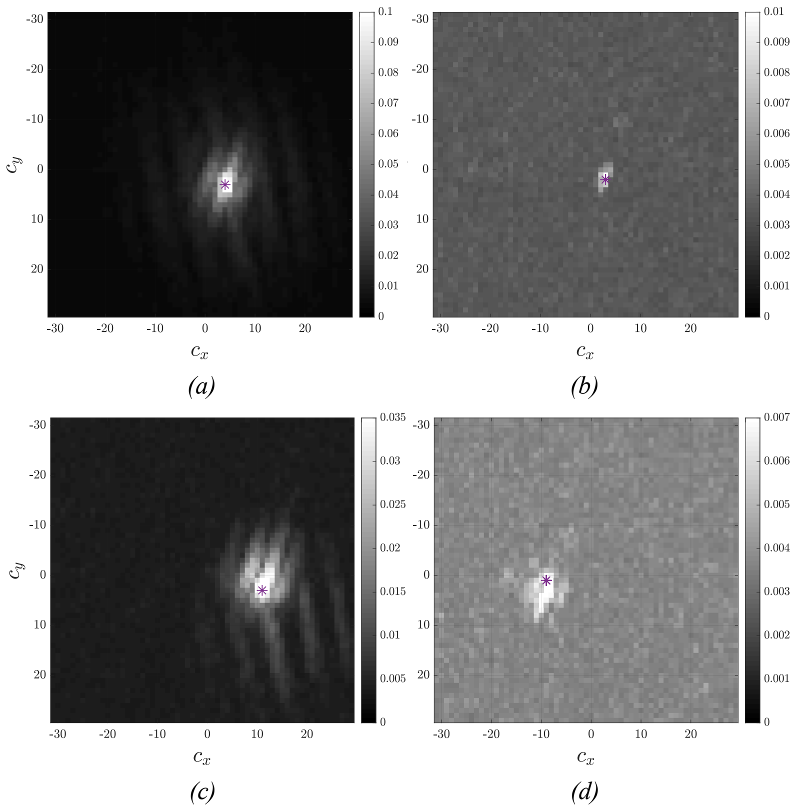


Figure 3.16: *Examples of behaviour of Φ_{peak} for different video frames, as a function of the horizontal and vertical shift. Specifically, (a) depicts the resulting function selecting $\mathbf{I}_f$ from a low-textured and static sequence; in (b), $\mathbf{I}_f$ comes from a still video in outdoor scenario; in (c), $\mathbf{I}_f$ is taken by a moving user, but in an almost flat scenario; in (d), $\mathbf{I}_f$ depicts an outdoor scenario with user motion. Symbol $*$ represents the global maximum position.*

some individuals are selected as parents of new children which are used for the next generation. Over subsequent generations, the algorithm converges toward the optimal solution.

After estimating the warping parameters $s_{\mathsf{fm}}, \alpha_{\mathsf{fm}}, \mathbf{c}_{\mathsf{fm}}$, last steps consist in:

i) compensate the scale, rotation and shift in $\mathbf{K}_{\mathsf{iv}}$ by means of the estimated transformation $\mathscr{T}_{\mathsf{s}_{\mathsf{fm}}\alpha_{\mathsf{fm}}\mathbf{c}_{\mathsf{fm}}}$;

j) resort to PCE as strategy for a correct device identification.

Precisely, exactly as reported in Section 3.2.3, we compute PCE between

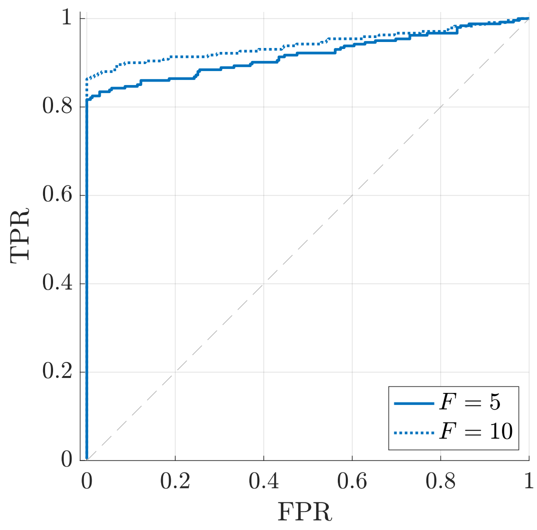


Figure 3.17: *ROC curves obtained testing $F = \{5, 10\}$ I-frames with the Fourier-Mellin-based strategy.*

$\mathbf{W}_f$ and the warped reference fingerprint $\mathbf{K}_{\mathsf{iv}}$. We come out with P_{fm}, defined as:

$$P_{\text{fm}} = \text{PCE}(\mathbf{W}_f, \mathscr{T}_{\mathsf{s}_{\text{fm}}\alpha_{\text{fm}}\mathbf{c}_{\text{fm}}}(\mathbf{K}_{\mathsf{iv}})). \tag{3.27}$$

Comparison with particle swarm results

In order to show the potentiality of Fourier-Mellin approach in dealing with source device identification problem, we apply the complete test strategy comparing the results with those achieved by particle swarm optimization. To this purpose, we select as device reference fingerprint $\mathbf{K}_d = \mathbf{K}_{\mathsf{iv}}$, and we pick exactly the same dataset used for testing the complete strategy with particle swarm. For each analyzed video frame, we estimate the relative shift between scaled and rotated $\mathbf{W}_f$ and $\mathbf{K}_{\mathsf{iv}}$ as shown in (3.25) , then we find scale and rotation following (3.26). Precisely, we set as search range for the mutual shift $\mathcal{C}_{\mathsf{fm}} = [-90, 90]$ both in horizontal and vertical directions. We verified this range includes all possible mutual shift between correlating video frames and reference fingerprint. Eventually, we evaluate P_{fm} as the PCE between the realigned fingerprint and $\mathbf{W}_f$.

For the sake of clarity, we use the very same accuracy metrics presented above, i.e., AUC and $\text{TPR}_{@0.01}$ of ROC curves. Precisely, Figure 3.17 shows ROC curves drawn testing 5 or 10 I-frames per video query. Notice that results maintain the previously derived trend: the more the tested frames, the better the attribution.

Table 3.4: AUC *and* $\mathrm{TPR}_{@0.01}$ *exploiting* $\mathbf{K}_{\mathsf{iv}}$ *as reference fingerprint, testing* $F = \{5, 10\}$ *random I-frames with the Fourier-Mellin-based strategy and particle swarm strategy.*

STRATEGY	$P_{\mathrm{fm}}, F = 5$	$P_{\mathrm{fm}}, F = 10$	$P_{\mathrm{comp}}, F = 5$	$P_{\mathrm{comp}}, F = 10$
AUC	0.92	0.94	0.96	0.97
$\mathrm{TPR}_{@0.01}$	0.82	0.87	0.87	0.91

For the sake of comparison, Table 3.4 shows the achieved AUC and $\mathrm{TPR}_{@0.01}$ using the Fourier-Mellin-based strategy and the particle swarm strategy presented in Section 3.2.3.

For both cases $F = 5, F = 10$, Fourier-Mellin approach presents worse performances with respect to the particle swarm strategy, even though reporting acceptable results which are always better than quick test method. Unfortunately, the required computational time increases. If particle swarm requires 57 seconds for testing one single query frame with the complete test strategy, the Fourier-Mellin approach implemented using the genetic algorithm needs 275 seconds for estimating the correct warping parameters and compute P_{fm}. This time growth is mainly due to Fourier transform computation, which needs a great number of samples to avoid undesired border effects and wrong parameters' estimation. Indeed, in order to obtain a good log-polar representation for Full-HD video sequences with pixel resolution 1080×1920, we exploit 4096×4096 Fourier transform samples. With respect to particle swarm which runs optimization on Full-HD sequences, the genetic algorithm suffers from this resolution enhancement and slow down temporal performances.

Nonetheless, the Fourier-Mellin approach presents interesting aspects as well. First of all, contrarily to particle swarm, it does not require a global optimization technique to find the correct scale and rotation parameters, because these can be directly derived from the position of the peak in phase correlation. This enables a faster estimation of the two parameters, in the order of few milliseconds in all the situations in which shift is known to be absent or negligible by construction. Furthermore, Fourier-Mellin does not require to fix search bounds for scale and rotation parameters. In principle, we could estimate every possible scaling factor and rotation angle, provided that Fourier transform is computed with a high resolution level.

The only obstacle preventing the use of closed form estimation consists in the presence of mutual shift between noise residuals and reference fingerprint. So far, it seems a hard task to find a relationship allowing the alignment (in terms of horizontal and vertical translation) between the log-polar representations of noise residual and device reference fingerprint, and this still remains an open problem. For this reason, Fourier-Mellin approach surely needs further investigations for solving the device identification problem in case of video stabilization. This strategy can potentially pave the way for accurate and fast solutions with respect to state-of-the-art. A first solution in this sense could be the use of ad-hoc sparse version of Fourier transform, enabling to quickly compute dense spectra at some frequencies of interest and disregarding other frequency ranges [99].

3.3 Detection and Localization of video temporal splicing

Solving source device identification on videos is just one of the primary steps of the work performed by the forensics investigator when he/she must assess the integrity of this kind of multimedia content. Indeed, there is an infinite list of possible forgeries and modifications videos can undergo after the acquisition. This is mainly due to last year developments in video editing software and computer graphics tools, where by manipulating visual contents has become a relatively easy task. Some forged videos are so well crafted to elude visual scrutiny even by forensic experts. For this reason, there is a growing interest for automatic tools which can reliably establish video integrity.

Within the large number of video forensics methods proposed in the literature, methods based on PRNU are independent of the specific type of manipulation, which is why they are drawing considerable attention in both research and applications. However, as previously stated, PRNU estimation is a much harder task for videos than for images, since videos are usually compressed with relatively low quality, compromising the sensor footprints. Another major problem is video stabilization, that causes misalignments of individual pixels across frames. Indeed, we shown in Section 3.2 that stabilization is a serious issue and PRNU-based methods are not effective anymore in the absence of suitable countermeasures.

Even neglecting the above problems, to reliably estimate the PRNU pattern, a large number of video frames should be available. Unfortunately, this is not always the case. Quite often, one is required to work in a *blind* setting, analyzing a single video of unknown origin downloaded from the net. In this situation, one can use some of the video frames to estimate the PRNU, but the quality will significantly impair, not only for the limited number of available frames but also because of their content correlation.

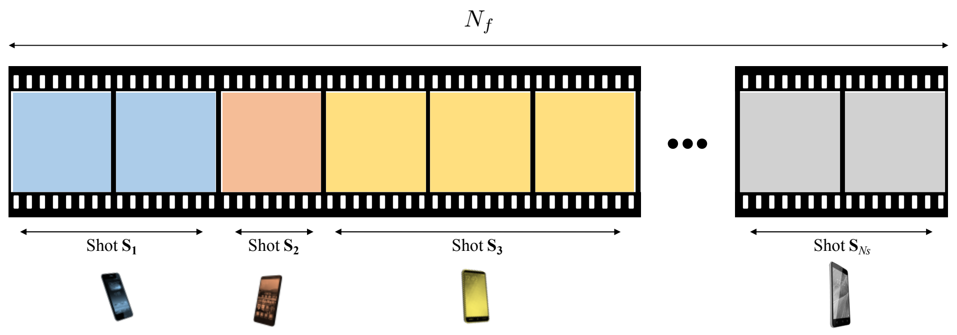


Figure 3.18: *Video sequence composed of N_s shots coming from unknown devices.*

A possible approach is to work on noise residuals estimated from the single video and extract as much information as possible from them. In [100] some initial video frames are used to estimate a reference pattern and check for video authenticity. In [101] the temporal correlation of noise residuals is analyzed through a Gaussian mixture model, while in [102] the inconsistencies of the photon shot noise characteristics are used for forgery detection.

In this section, we address the detection of video temporal splicing, which arises when two or more video shots are used to compose a new video. As in [100], the noise residuals of the initial video frames are used to extract a reference pattern (a coarse PRNU estimate), which is used in turn to detect the presence and position of a possible splicing. The process is then iterated, with the aim to detect eventually the precise combination of different shots and their temporal composition. To the best of our knowledge, this is the first time this manipulation is considered in the literature. In the following, we introduce the goal and show the proposed solution.

3.3.1 Problem formulation

We aim at detecting and localizing video temporal splicing in a blind way. Specifically, we do not know how many source devices are spliced together, nor the number of splicing portions and the temporal position of the splicing. Formally, let us consider a video $\mathbf{V}$ modeled as the temporal concatenation of an unknown number N_s of shots S_n, $n \in [1, N_s]$, i.e., $\mathbf{V} = \{\mathrm{S}_1, \mathrm{S}_2, ..., \mathrm{S}_{N_s}\}$. Each shot S_n is composed by an unknown number of frames recorded from a single device. Devices are assumed to be unknown.

Figure 3.18 shows an example of video compilation composed by N_s shots. Our goal is to estimate the amount of shots N_s, and segment the video $\mathbf{V}$ into its originating shots S_n. In order to split the video into the set of originating shots, we leverage PRNU-based source device attribution. In principle, if each shot corresponds to a single acquisition device whose PRNU is known, it could be possible to aggregate frames sharing significant correlation with each PRNU, thus detecting and localizing the various splices. However, we consider the challenging scenario in which shots' PRNUs are not available, as we do not know the camera models exploited for generating the video compilation under analysis. To overcome this problem, we propose an algorithm to estimate the various camera fingerprints directly from video frames, in an iterative fashion. This allows to blindly identify how many shots generate the compilation, and localize the splicing portions.

For the sake of clarity, we suppose to work with video compilations composed by non-stabilized sequences. Nonetheless, in section dedicated to results, we provide experiments on both non-stabilized and stabilized videos. In the next following, we report a detailed description of the pipeline.

3.3.2 Proposed method

As previously stated, every analyzed video $\mathbf{V}$ is the combination of various shots with distinct characteristics. More specifically, we are completely unaware of the number of involved devices, related camera models, and number of frames of each splicing shot. Here we show the rationale driving the proposed method through an example, followed by an exhaustive description of the algorithm.

Let us suppose we randomly select from the whole sequence a reference frame $\mathbf{I}_r$ and extract its noise residual $\mathbf{W}_r$, as proposed in [10]. This frame belongs to a random shot, thus its noise $\mathbf{W}_r$ is supposed to correlate only with noise residuals extracted from other frames of the same shot. By scanning all video frames $\mathbf{I}_f$, $f \in [1, N_f]$ and extracting the relative noise $\mathbf{W}_f$, we define the cumulative sample mean noise as

$$\overline{\mathbf{W}}(f) = \sum_{i=1}^{f} \frac{\mathbf{W}_i}{f}. \tag{3.28}$$

This cumulative average noise contains information about noises extracted from all frames until the f-th one. If $\mathbf{V}$ is generated from a single shot, $\overline{\mathbf{W}}(f)$ is directly related to the PRNU of the recording camera as defined in (2.1). In case of multiple shots, $\overline{\mathbf{W}}(f)$ contains averaged information about different shots' fingerprints, depending on f.

Taking into account these considerations, we can solve the source device identification problem between the available fingerprint estimate $\overline{\mathbf{W}}(f)$ and

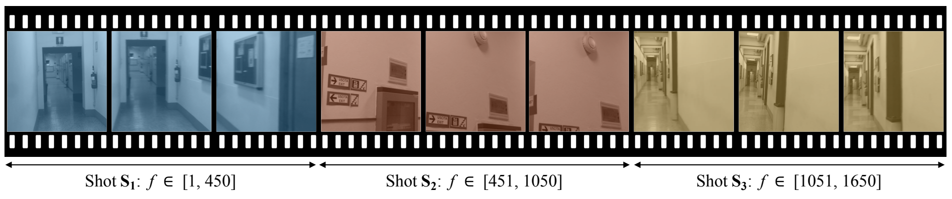

Shot $\mathbf{S}_1$: $f \in [1, 450]$
Shot $\mathbf{S}_2$: $f \in [451, 1050]$
Shot $\mathbf{S}_3$: $f \in [1051, 1650]$

Figure 3.19: *Video sequence composed of* 3 *shots coming from* 3 *unknown devices.*

the reference frame $\mathbf{I}_r$. It is worth noticing that computing the frame-variant NCC denoted as $c(f) = \text{NCC}(\mathbf{W}_r, \overline{\mathbf{W}}(f) \circ \mathbf{I}_r)$, we can observe this behavior:

- If $f < r$ and the considered f frames do not belong to the same shot of $\mathbf{I}_r$, then $c(f)$ is low and more or less constant. As a matter of fact, $\overline{\mathbf{W}}(f)$ is a completely wrong estimate of the fingerprint related to the reference shot and does not correlate with $\mathbf{W}_r$.
- At a given $f \leq r$, $\overline{\mathbf{W}}(f)$ starts being built exploiting noise residuals from frames belonging to the very same device of $\mathbf{I}_r$. Hence, $\overline{\mathbf{W}}(f)$ starts matching $\mathbf{W}_r$, and $c(f)$ begins to increase.
- After all frames of the reference device have been scanned (i.e., the f-th and r-th frames come from different devices), $c(f)$ starts dropping, since $\overline{\mathbf{W}}(f)$ begins containing contributions from noises not correlating anymore with $\mathbf{W}_r$.

For the sake of clarity, Figure 3.19 reports a compilation composed by three splicing portions:

- S_1, composed by frames $\mathbf{I}_f$, $f \in [1, 450]$;
- S_2, composed by frames $\mathbf{I}_f$, $f \in [451, 1050]$;
- S_3, composed by frames $\mathbf{I}_f$, $f \in [1051, 1650]$.

The behaviour of the relative $c(f)$ is depicted in Figure 3.20 .

If the reference frame is $\mathbf{I}_{100}$ (i.e., belonging to $\mathbf{S}_1$), $c(f)$ increases up to $f = 450$, then it starts dropping as frames after $\mathbf{I}_{450}$ do not belong to S_1 anymore. If the reference frame is $\mathbf{I}_{700}$ (i.e., belonging to $\mathbf{S}_2$), $c(f)$ is almost flat for $f \leq 450$ (i.e., frames belonging to $\mathbf{S}_1$), shows an increasing behavior for $450 < f \leq 1050$ (i.e., frames belonging to $\mathbf{S}_2$), then it drops again for $f > 1050$ (i.e., frames belonging to $\mathbf{S}_3$). A coherent behavior can be observed if we consider reference frame $\mathbf{I}_{1300}$.

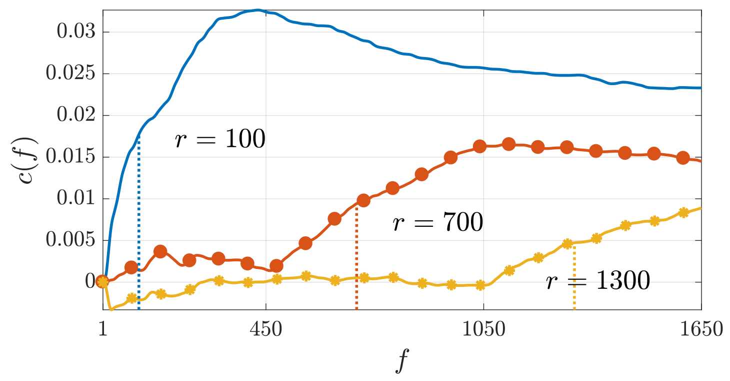


Figure 3.20: *Example of* $c(f)$ *behaviour over the video shown in Figure 3.19.*

Bearing this in mind, the proposed pipeline for blind detection and localization of temporal splicing consists of the following steps:

a) selecting the reference frame – randomly select one reference frame from the video and compute $c(f), f \in [1, N_f]$;

b) clustering frames – group together frames for which $c(f)$ locally increases and delete the selected group from the entire video;

Iterate steps *(a)* and *(b)* until almost all video frames have been clustered in different groups;

c) clustering shots – to counteract the problem of over-estimating the number of splicing shots, cluster the groups of frames with higher inter-correlation;

d) assigning left-out frames – assign the remaining frames to the best-matching shot.

It follows an exhaustive description of each step.

(a) Selecting the reference frame

Since information about temporal segmentation is not available, the only way for selecting the reference frame is to pick it up randomly. Actually, interpretation of $c(f)$ is not always straightforward like in Figure 3.20. As a matter of fact, correlation $c(f)$ can exhibit an increasing behavior even for frames not belonging to the same shot of $\mathbf{I}_r$, as well as multiple local maxima (e.g., due to correlated frame content). Therefore, to increase the algorithm's robustness, we perform multiple experiments, picking up a pool of different reference frames.

The algorithm extracts R possible $\mathbf{I}_r$ frames, and computes $c_r(f)$ for each realization $r \in [1, R]$. We define three quantities useful to evaluate $c_r(f)$ goodness:

- The maximum value of $c_r(f)$, defined as
$$\mathrm{M}_r = \max_f \left(c_r(f)\right) . \tag{3.29}$$
- The frame index related to the maximum $c_r(f)$ value, defined as
$$m_r = \arg\max_f \left(c_r(f)\right) . \tag{3.30}$$
- The largest set of frame indexes for which $c_r(f)$ shows a monotonically increasing behavior, defined as Δ_r.

The best reference $\tilde{r}$ out of the R ones is selected as the realization with highest M_r, given that $m_r \in \Delta_r$. This ensures that frames whose index lies in Δ_r belong to a single device.

(b) Clustering frames

Once the best realization $\tilde{r}$ has been selected, we average noise residuals of frames belonging to $\Delta_{\tilde{r}}$, in order to estimate a fingerprint $\hat{\mathbf{K}}_n$ which will be related to a new shot $\hat{\mathbf{S}}_n$.

To cluster frames together, we follow the standard PRNU-based source attribution pipeline: being $\hat{\mathbf{K}}_n$ the estimated fingerprint, noises from all video frames are correlated with $\hat{\mathbf{K}}_n$. We assign to the new shot $\hat{\mathbf{S}}_n$ all frames for which NCC is above a predefined threshold. Next operation consists in removing the estimated group of frames from the video sequence, and iterate steps *(a)* and *(b)* until remaining frames are less than a default value (i.e., 100 in our experiments).

(c) Clustering shots

Estimation of true fingerprint from a small subset of frames is far from being an easy task. For this reason, it sometimes happens that frames belonging to the same original shot are not clustered together, due to low correlation values. Therefore, we usually end up with an estimated compilation $\hat{\mathbf{V}} = \{\hat{\mathbf{S}}_1, \hat{\mathbf{S}}_2, ..., \hat{\mathbf{S}}_{M_s}\}$, whose number of shots is higher than the true one (i.e., $M_s > N_s$).

Some control on over-estimation is thus necessary. On the other hand, it is still better over-segmenting the compilation than clustering shots of different sources. To this purpose, we propose a clustering strategy for blindly grouping shots wrongly split:

- We compute the reference noise pattern $\hat{\mathbf{K}}_n$ for each estimated shot in $\hat{\mathbf{V}}$.
- We correlate through NCC all pairs of reference noise patterns.
- We cluster different shots if and only if each shot of the cluster has pairwise NCC with *all* other shots greater than a threshold Γ, and the cluster is composed by temporally adjacent shots.
- The estimated video sequence $\hat{\mathbf{V}}$ now includes a reduced set of shots, whose fingerprints are the average of noise patterns inside the same cluster.

This procedure is iterated manifolds, until no more shots are aggregated.

***(d)* Assigning left-out frames**

At this step, the compilation $\hat{\mathbf{V}} = \{\hat{\mathbf{S}}_1, \hat{\mathbf{S}}_2, ..., \hat{\mathbf{S}}_{L_s}\}$ includes the reduced set of L_s shots, while at most 100 remaining frames have not been assigned to any contribution. The easiest way for labeling them is to apply the standard PRNU-based source attribution pipeline. We assign each singleton frame to the shot whose estimated fingerprint better correlates with the frame noise residual.

3.3.3 Experiments and validation

In this section we first introduce the datasets used for experiments, then we report the achieved results.

Datasets

Splicing portions have been collected from the same dataset used for experiments in Section 3.2.5, the Vision dataset [29]. More specifically, we created two distinct datasets for non-stabilized and stabilized compilations. For the sake of brevity, from now on we describe the generation process of non-stabilized compilations, but procedure still remains the same for both cases.

From dataset [29], we only select non-stabilized devices with minimum Video Resolution set to HD-Ready ($\mathrm{VR} \geq 720p$). Then, 5 videos per device are collected, randomly picking from indoor/outdoor scenarios and considering only move/panrot acquisition modes. This choice comes from the idea of generating plausible results, since combinations of flat or static videos are actually less likely to be found.

For each device, we cut the 5 selected videos at frame index 150, 300, 450, 600, 750, respectively, in order to generate splicing portions of different lengths. The resulting splicings are then cropped to common resolution of 720×720 pixels and gray-scale converted. Since the available non-stabilized devices are 19, we end up with a pool of 95 distinct splices.

The final video compilation is obtained as the temporal concatenation of $N_s \in [3, 6]$ splicing portions, randomly extracted from the pool. Following this pipeline we generated two datasets, covering 150 non-stabilized videos and just as many stabilized.

Evaluation Metrics

We developed two kinds of accuracy measures for inferring the quality of proposed method in splicing localization. Specifically, A_{orig} and A_{est} are defined as:

- A_{orig}: for each *original* shot, A_{orig} is the percentage of frames belonging to that shot which actually have been labeled as a unique cluster in the estimation process. This measure detects presence of over-segmentation.
- A_{est}: for each *estimated* shot, A_{est} is the percentage of frames belonging to that shot, which effectively belongs to a unique original splice. It decreases in case of under-segmentation.

Concerning quality evaluation in identifying the number of shots in the compilation, it is paramount to take into account previous considerations made in Section 3.3.2. Our goal is to reduce as much as possible the error in the amount of estimated shots, still favoring over-segmentation in order not to mix various devices together. We define E_s as the error in estimating the number of shots which generates the video.

Accuracies and E_s are averaged over the total amount of contributions in a single compilation.

Results

We show results in terms of mean $\mathrm{A}_{orig}, \mathrm{A}_{est}, \mathrm{E}_s$ over the two datasets of non-stabilized and stabilized videos. More specifically, we evaluate these measures for different values of threshold Γ exploited for clustering splices.

Figure 3.21 depicts outcomes for non-stabilized compilations. The more Γ increases, the less splices are clustered. Hence, while accuracy A_{orig} decreases for over-segmentation, A_{est} gets approximately to 0.95. Note that we must necessarily investigate accuracy behavior together with E_s,

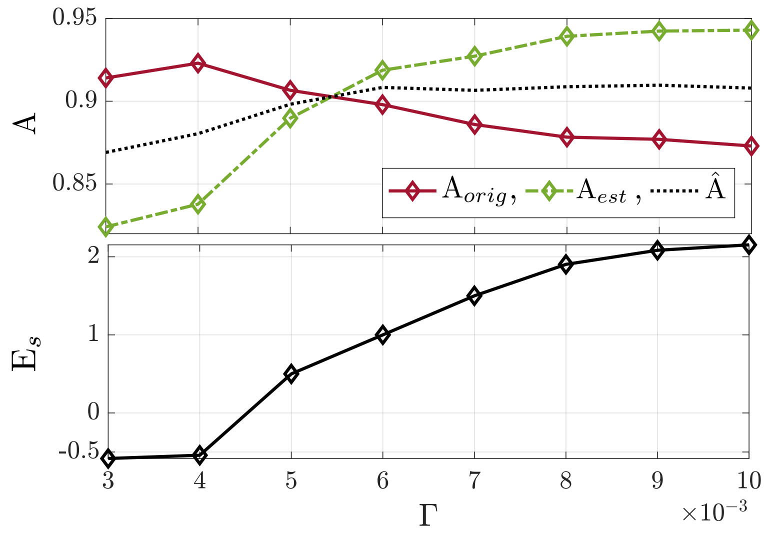


Figure 3.21: *Results for non-stabilized video compilations.*

otherwise we could fall into several interpretation mistakes. Higher values for A_{est} are feasible only as long as E_s does not excessively grow. For this reason we introduce a new accuracy measure, averaging A_{orig} and A_{est} versus Γ, ending up with $\hat{A}$.

We think of $\hat{A}$ representing a good measure for the selection of best Γ for clustering. Indeed, $\hat{A}$ takes into account both over-segmentation risk (highlighted by A_{orig}) and under-segmentation risk (stressed by A_{est}). In light of this, we note that the best threshold values are $\Gamma = \{5, 6\} \times 10^{-3}$, which guarantee $\hat{A}$ around 0.9, and segmentation error E_s below $+1$ over-estimated splices on average.

As far as stabilized compilations are concerned, results are shown in Figure 3.22. Note that, in this situation, the PRNU-based source attribution approach is severely hindered. As a matter of fact, A_{orig} is always well below 0.7. On the other hand, A_{est} achieves very good measures, reaching scores about 0.94. We must beware of this result: the behavior of E_s is far from being acceptable, as $E_s > 4$ for all thresholds. This means that we are actually over-segmenting shots very often. However, as PRNU estimation is known to be a challenging task for stabilized videos, these results are expected.

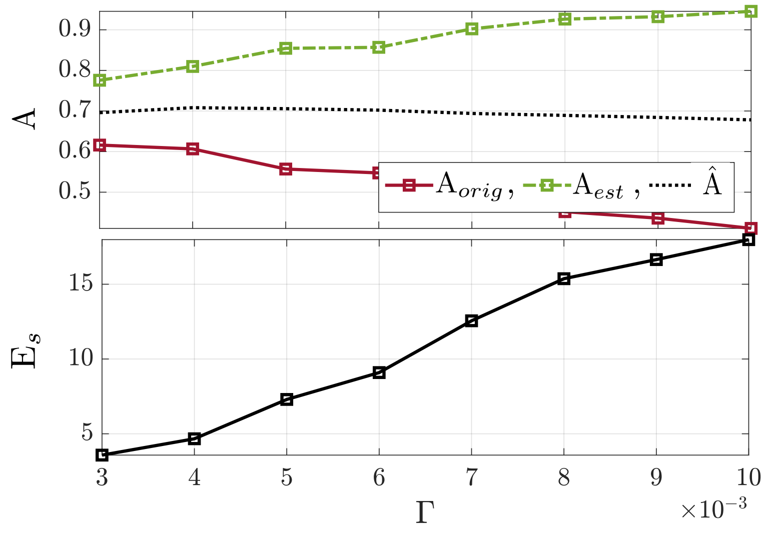


Figure 3.22: *Results for stabilized video compilations.*

3.4 Conclusions

In the initial part of this chapter, we propose several solutions to the problem of video source attribution when motion stabilized video sequences are considered. The experimental campaign is conducted on a publicly available dataset composed by almost 400 videos coming from stabilized and non-stabilized mobile devices. The best results for video attribution are obtained if images are available for reference fingerprint estimation, whereas using only videos worsen the achieved performance as expected. It is worth noting that video stabilization is performed directly on-board by proprietary software, and we have no control over it, thus making the experiments completely realistic. Despite this lack of knowledge, modeling video stabilization with similarity transformation proves to be quite effective. Overall, it is required less than a minute per frame for parameters estimation by means of particle swarm optimization.

Additionally, we investigate an alternative strategy to estimate the similarity parameters based on Fourier Mellin transform. Though this methodology still needs improvement in terms of asked computation time and storage, it can potentially pave the way for efficient transformation estimations which do not require global optimization strategies.

Eventually, we highlight the effect of using the first acquired video frame for camera attribution. As this frame is often not stabilized, including it within the experimental campaign can produce misleading results and leads to wrong conclusions. This is especially true if we consider a future scenario in which mobile devices will start recording videos even before pressing the rec button (e.g., as already proposed in the latest Android-based Google devices). Indeed, in this situation, the concept of first acquired frame becomes fuzzy, and possibly all available frames can be motion compensated.

As a further analysis on video sequences, we consider the problem of blind detection of video splicing exploiting PRNU-related traces. We investigate video compilations composed by an unknown number of shots coming from an unknown amount of different devices. The proposed algorithm estimates the number of shots as well as their starting and ending points in time. We leverage the idea that noise traces (related to PRNU) extracted from different frames within a sequence should correlate only if frames have been acquired with the same device. It is therefore possible to iteratively group frames generated from the same device, eventually estimating all frame clusters, i.e., different shots.

We test our method on the same dataset used for source device identification. Despite the promising results obtained on non stabilized video sequences, this study confirms that video stabilization is a highly-corruptive operation in terms of PRNU-based detectors. Whenever a compilation of stabilized sequences is under investigations, the issues due to the risible amount of video frames and frame misalignment require suitable countermeasures similar to those previously presented and additional investigations as well.

CHAPTER 4

Source device anonymization

In this chapter, we present all the investigations regarding counter-forensics methods, namely strategies to undermine performances of forensics algorithms. Specifically, we focus on counter-forensics methods related to the source device identification problem. In this vein, the goal is now reversed with respect to previous tasks and consists in source device anonymization. A given visual content is manipulated in such a way it cannot be linked anymore to its provenance device. As previously reported, the motivations behind source device anonymization are basically two: spotting the weaknesses of source device identification algorithms and preserving the privacy of data owners.

Since this is a relatively new field of study, our research focuses on image anonymization. To this purpose, we propose two strategies. The former one deletes a predefined set of pixels and inpaints them from neighbours exploiting regularization techniques. This operation helps to reduce the PRNU traces left on the image, lowering the cross-correlation test with the candidate device while guaranteeing an acceptable image visual quality. The latter strategy leverages recent findings in the deep learning field to attack PRNU-based detectors. Specifically, we focus on the possibility of editing an image through convolutional neural networks (CNNs) in a visually imperceptible way, still hindering PRNU noise estimation.

In the following lines, we first report the state-of-the-art on image anonymization, later we show the novelties of our proposals.

4.1 State-of-the-art in image anonymization

In the latest years, the increased privacy concerns make the need for effective anonymization methods rather pressing. Furthermore, studying the boundaries of image anonymization can enable analysts to be aware of the robustness of camera attribution methods in the presence of malicious attacks. For these reasons, device anonymization techniques tailored to remove or hinder PRNU traces have been developed.

Among the developed techniques, some require the knowledge of the PRNU pattern to be deleted. As an example, the authors of [103] and [104] propose different iterative solutions to delete a known PRNU from a given picture. They propose to remove the PRNU by means of the subtraction of this from the image intensity.

Other methods work by blindly modifying pixel values and scrambling their positions in order to make the underlying PRNU unrecognizable. Indeed, they do not require the knowledge of the PRNU pattern, thus being more suitable in cases where the corresponding device is not available.

For instance, [105] investigates the robustness of PRNU against standard processing operations like denoising, JPEG compression, re-compression and demosaicking. They show that even after eight denoising steps there are issues in anonymizing images, because the correlation between the image noise residual and PRNU is still significant. Furthermore, they demonstrate that compression or demosaicking operations contribute in lowering the accuracy of source device attribution, but cannot remove the PRNU pattern in an acceptable manner.

Alternatively, [106, 107] propose to anonymize images by misaligning the noise residuals and the PRNU pattern. As a matter of fact, as seen with video stabilization, in order to compute the correlation test, the noise content of both terms should be aligned in pixels. In order to hinder the attribution performances, the authors show that neither geometric transformations are enough but we need to incorporate irreversible transformations. They propose to anonymize images by applying seam-carving to change pixel locations, thus introducing irreversible misalignments.

More recently, [92] compares patch-based methods to shuffle small image blocks. In their paper, authors explore the natural self-similarity of digital images to move small patches into different positions of the image, thus impeding the pixel alignment between noise residual and device fingerprint.

4.2 Inpainting-based image anonymization

In this section, we investigate parallel and fast inpainting techniques as methods for image anonymization. Inpainting is a well-known topic, which refers to the application of simple or sophisticated algorithms for reconstructing lost or corrupted portions of an image (e.g., by solving partial differential equations, by applying sparse domain transformations or regularization of inverse problems, etc.). The rationale behind our approach is that, by deleting and reconstructing each pixel from its neighbours, PRNU effect can be strongly attenuated. The proposed method mainly works in two steps:

1. each image pixel is substituted by its inpainted value, in order to corrupt the PRNU;
2. pixels around edges are replaced by denoised versions of the original ones in order to mask possible visual artifacts around sharp discontinuities.

Even though the literature is wide and provides many advanced solutions, we investigate the use of simple yet effective inpainting schemes that keep computational complexity at bay. Specifically, we only consider solutions that reconstruct pixels by solving inverse regularized problems.

In particular, it is important to point out that a regularization which perfectly reconstructs the original image would lead to a high visual quality result, but would be totally useless for what concerns the anonymization task, as the traces of PRNU would remain almost unchanged. Therefore, in order to achieve our goal, the algorithm proposes a trade off between quality and anonymization of the outputs.

4.2.1 Proposed method

The problem of image anonymization translates into the following constraint: given an image $\mathbf{I}$, attenuate its PRNU traces so that it is impossible to attribute the image to the device that shot it. In other words, if $\mathbf{I}$ is acquired with a device whose estimated PRNU is $\mathbf{K}_d$, it is reasonable that its $\mathrm{NCC}(\mathbf{W}, \mathbf{I} \circ \mathbf{K}_d) > \Gamma$, as previously shown in Section 2.2. Our goal is to modify $\mathbf{I}$ through editing techniques in order to obtain an anonymized version $\tilde{\mathbf{I}}$ with the following properties:

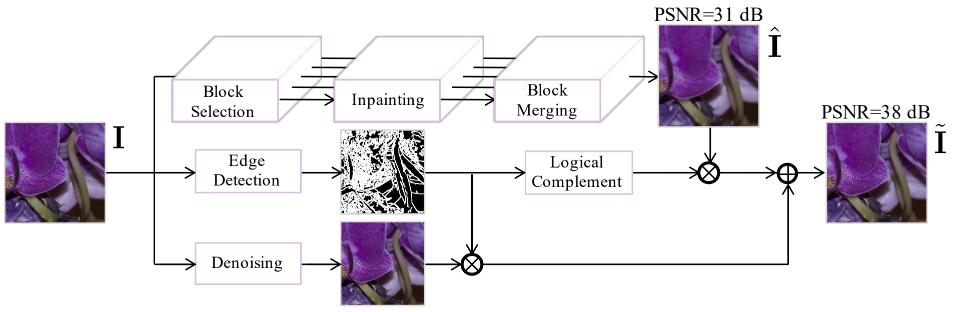


Figure 4.1: *Pipeline of the proposed anonymization strategy: the original image* $\mathbf{I}$ *undergoes block selection, and every modified version is inpainted in parallel. Results are merged to obtain* $\hat{\mathbf{I}}$*, which is further processed around edges, thus obtaining the anonymized image* $\tilde{\mathbf{I}}$*.*

- NCC between the noise extracted from the output image $\tilde{\mathbf{I}}$ (i.e., $\tilde{\mathbf{W}}$) and the camera PRNU $\mathbf{K}_d$ gets to reasonably low values, therefore $\text{NCC}(\tilde{\mathbf{W}}, \tilde{\mathbf{I}} \circ \mathbf{K}_d) < \Gamma$;
- image $\tilde{\mathbf{I}}$ is not visually corrupted by the applied editing operations (i.e., peak-signal-to-noise ratio between $\mathbf{I}$ and $\tilde{\mathbf{I}}$ assumes high values).

The proposed pipeline, shown in Figure 4.1, is the following:

1. select and delete different blocks of $\mathbf{I}$;
2. blocks of $\mathbf{I}$ are processed in parallel, reconstructing the erased pixels from neighboring ones as in a classical inpainting problem;
3. image $\hat{\mathbf{I}}$ is reconstructed by splicing together all the inpainted blocks;
4. the final anonymized image $\tilde{\mathbf{I}}$ is obtained by further processing pixels around edges of $\hat{\mathbf{I}}$, in order to mitigate inpainting artifacts near sharp discontinuities and increase visual quality.

The algorithm is applied separately on each color plane. In the following, we describe each step focusing on a single color plane, without loss of generality.

Block selection

The first step consists in selecting and deleting different image blocks as shown in Figure 4.2. The selection of pixels is performed through N pixel selectors $\mathbf{S}_s$, $s \in [1, N]$, applied in an element-wise fashion to $\mathbf{I}$. Specifically, each pixel selector $\mathbf{S}_s$ is a matrix with the same size of the image, set

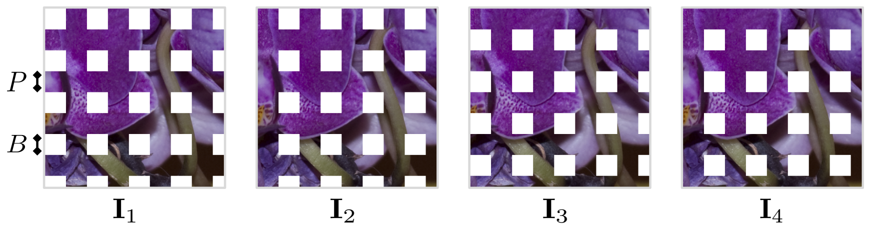


Figure 4.2: *Results of applying 4 pixel selectors to image* $\mathbf{I}$*: each selector* $\mathbf{S}_s$ *selects and deletes different blocks of the original image. White areas represent deleted pixels.*

to 0 if the pixel must be removed and to 1 elsewhere. This matrix, if multiplied pixel-wise with image $\mathbf{I}$, selects and erases areas of $B \times B$ pixels, interleaved in both horizontal and vertical directions by a fixed gap of P pixels left at their original values. In order to gradually select and cancel all image pixels, each selector $\mathbf{S}_s$ is orthogonal to the others. This means that each selector deletes a different portion of the image, but the whole image is covered considering the effect of all selectors eventually.

Figure 4.2 shows an example in which 4 pixel selectors are multiplied by the original image in order to cancel some specific regions. We define each modified version of $\mathbf{I}$ as $\mathbf{I}_s = \mathbf{S}_s \circ \mathbf{I}$, with $s \in [1, N]$.

Image inpainting

We investigate the family of inpainting methods based on the solution of simple ℓ_1 and ℓ_2 regularized inverse problems. Our choice is motivated by the fact that this class of problems has been broadly implemented through standard and efficient iterative methods [108].

In this scenario, the inpainted image $\hat{\mathbf{I}}_s$ is estimated by solving the minimization problem:

$$\hat{\mathbf{I}}_s = \arg\min_{\bar{\mathbf{I}}_s} \; \|\mathbf{S}_s \circ \bar{\mathbf{I}}_s - \mathbf{I}_s\|_{\mathrm{F}}^2 + \mu \, \|R\{\bar{\mathbf{I}}_s\}\|_p^p \, . \tag{4.1}$$

The pixel selector $\mathbf{S}_s$ selects the pixels from $\hat{\mathbf{I}}_s$, $\|\cdot\|_{\mathrm{F}}$ represents the Frobenius norm, μ is the penalty weight associated to the regularizer operator $R\{\cdot\}$, and $\|\cdot\|_p$ is the entrywise ℓ_p norm. The first term of the objective function (i.e., $\|\mathbf{S}_s \circ \bar{\mathbf{I}}_s - \mathbf{I}_s\|_{\mathrm{F}}^2$) is a fitting condition. It basically imposes that the inversion result approximately honors the known samples of the image. The second term (i.e., $\mu \, \|R\{\bar{\mathbf{I}}_s\}\|_p^p$) is the regularizer, which provides the condition to be respected by the inpainted pixels. Different regularizing operators and norms lead to different inpainting results.

A reasonable regularizer condition consists in imposing some smoothness constraints (i.e., $R\{\cdot\}$ should be a roughening operator). This ensures inpainted values to be not too dissimilar from neighboring pixels, which is desirable especially in flat regions of natural images (i.e., far from edges). Among the many regularization operators available in the literature, we make use of $R\{\cdot\} = \sqrt{(D_x\{\cdot\})^2 + (D_y\{\cdot\})^2}$, where operations are applied element-wise and $D_x\{\cdot\}$ and $D_y\{\cdot\}$ are the horizontal and vertical derivative operators, respectively.

For instance, the result of applying $D_x\{\cdot\}$ to an image $\bar{\mathbf{I}}_s$ is defined as

$$[\bar{\mathbf{I}}_{s_x}]_{i,j} = \begin{cases} [\bar{\mathbf{I}}_s]_{i,j} - [\bar{\mathbf{I}}_s]_{i,j+\Delta} & j \in [1, H - \Delta] \\ 0 & j \in [H - \Delta + 1, H] \end{cases}, \tag{4.2}$$

where $\Delta > 0$ is the desired pixel gap for the derivative calculation, H represents the image height and width, as we are dealing with square images, and (i, j) represent the pixel locations within the image. By setting different gaps Δ, various definitions of derivative can be used.

For what concerns the norm applied to the regularization term, we consider two strategies:

1. a ℓ_2 norm leading to the well-known Tikhonov regularization [109], which is the most widespread technique for inverting ill-conditioned problems;
2. a ℓ_1 norm strategy, known as Total Variation (TV) regularization [110], which exhibits edge preserving properties.

While the ℓ_2 strategy is very simple from an implementation point of view, it is known to introduce an overall smoothing effect that may be undesirable around sharp edges. For this reason we also investigate the ℓ_1 norm applied to the previously defined operator, which is widely used for preserving edges and discontinuities in the final inpainting solution.

Block Merging

After obtaining the inpainted versions $\hat{\mathbf{I}}_s$ of each image $\mathbf{I}_s$, we construct the image $\hat{\mathbf{I}}$ by merging the inpainted blocks:

$$\hat{\mathbf{I}} = \sum_{s=1}^{N} (1 - \mathbf{S}_s) \circ \hat{\mathbf{I}}_s. \tag{4.3}$$

Note that each pixel of $\hat{\mathbf{I}}$ is an inpainted pixel, and no original pixels of $\mathbf{I}$ survive this operation. Moreover, due to $\mathbf{S}_s$ definition, each pixel of $\hat{\mathbf{I}}$ is reconstructed from a single $\hat{\mathbf{I}}_s$.

Edge Processing

Inpainting applied with the aforementioned solutions may still introduce some undesirable visual artifacts around edges (the effect can be noticed in Figure 4.1, where image $\hat{\mathbf{I}}$ has low peak signal-to-noise ratio). Therefore, to increase image visual quality, a further edge processing operation is applied. More specifically, we substitute pixels of $\hat{\mathbf{I}}$ around image edges with pixels coming from a denoised version of $\mathbf{I}$, in order to avoid the reintroduction of too much PRNU information. Indeed, [105] shows that denoising contributes in attenuating PRNU, thus motivating our approach.

Formally, the edge reconstruction pipeline is as follows:

1. the original image $\mathbf{I}$ is denoised using two successive steps of BM3D algorithm [111](with variance parameter $\sigma = 7$), thus obtaining $\dot{\mathbf{I}}$;
2. the edges of $\mathbf{I}$ are extracted using Canny edge detector (with its default parameters defined in [112]), obtaining a binary edge mask $\mathbf{E}$ (the same size of $\mathbf{I}$), which is 1 only at edge locations and 0 elsewhere;
3. edge-mask $\mathbf{E}$ is dilated by means of a disk structural element (with radius of 3 pixels);
4. the final anonymized image is defined as $\tilde{\mathbf{I}} = \hat{\mathbf{I}} \circ (1 - \mathbf{E}) + \dot{\mathbf{I}} \circ \mathbf{E}$.

Figure 4.1 shows the effect of applying edge processing to $\hat{\mathbf{I}}$. Note that peak signal to noise ratio (PSNR) of $\tilde{\mathbf{I}}$ is increased by $7\,\mathrm{dB}$. Moreover, the computational complexity of this step is mainly due to denoising operation, which is almost negligible with respect to the rest of the inpainting pipeline.

4.2.2 Results

In this section we report all the details about the conducted experimental campaign. To this purpose, we first introduce the used experimental methodology. Then, we report results obtained with different inpainting strategies.

Methodology

Our experimental setup mimics that of Entrieri *et al.* [92], one of the most recent image anonymization approaches. For this reason, as image dataset, we randomly select 600 never-compressed Adobe Lightroom images from the Dresden Image Database [42] coming from 6 camera instances (Nikon D70, Nikon D70s, Nikon D200, two devices each). All images are synchronized to landscape orientation and cropped to the central portion of size 512×512 pixels.

The estimation of the clean sensor fingerprint $\mathbf{K}_d$ for each device is obtained from 25 homogeneously lit flatfield images as typically suggested [20, 113]. Image noise residuals $\mathbf{W}$ are computed with the Wavelet-based filter [114] commonly used in PRNU extraction [20].

We show results in terms of receiver operating characteristic (ROC) curves of a camera identification PRNU-based detector. Specifically, fixing the PRNU of a given camera, NCC obtained from anonymized images taken with that camera define the set of positive samples, whereas the set of negative ones includes NCC values from all images not taken with that camera. Our goal is to reduce as much as possible the area under the curve (AUC), thus making the PRNU-based detector not working.

Moreover, to evaluate image quality level, we report the relationship between PSNR and true positive rate (TPR) calculated at a fixed false positive rate (FPR) of 1%, which we denote as $\mathrm{TPR}_{@0.01}$. The goal is to reach a high PSNR with low values of $\mathrm{TPR}_{@0.01}$ and AUC. Results are always averaged on all 6 devices.

To distinguish between the proposed inpainting strategies, we use the following notation: each technique is identified by the label $\ell_p^{(B)}$, where ℓ_p represents the selected norm for the cost function regularization (i.e., $p \in \{1, 2\}$), while the superscript defines the size of the $B \times B$ regions deleted by the pixel selectors.

For what concerns the derivative implementation, we noticed that $\Delta = 3$ provides a good trade off between high reconstruction quality and low PRNU correlation. Indeed, smaller Δ tend to inpaint the image from nearer (and more correlated) pixels, consequently enhancing the correlation with the camera fingerprint. Conversely, larger Δ result in low PSNR. The penalty weight associated to the regularizer is $\mu = 10^{-20}$ for both ℓ_2 and ℓ_1 norms, since higher values oversmooth the image, thus reducing the visual quality.

Testing

The first experiment aims at showing that the proposed pipeline is actually able to fool PRNU-based camera attribution detectors. To this purpose, we tested both ℓ_2 and ℓ_1 inpainting by considering different block sizes $B \in \{3, 5\}$. Figure 4.3 reports promising ROC curves, as their slopes are approximately 45° degrees (i.e., perfect anonymization). Every strategy seems to provide satisfying performances, even though, the bigger the block dimensions ($B = 5$ instead of 3), the better the anonymization results.

For what concerns the inpainting effect on image quality, Figures 4.4

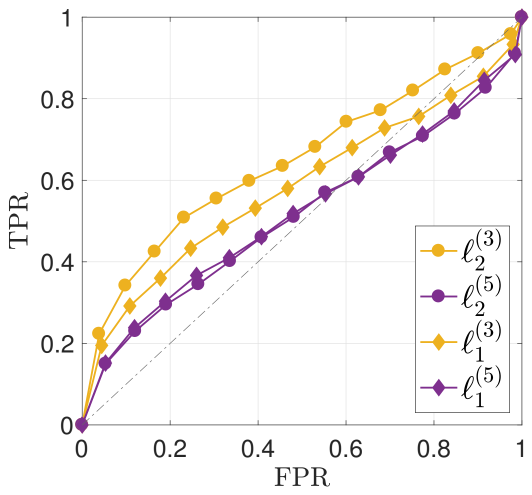


Figure 4.3: *ROC curves after camera anonymization process (each color represents a different inpainting strategy).*

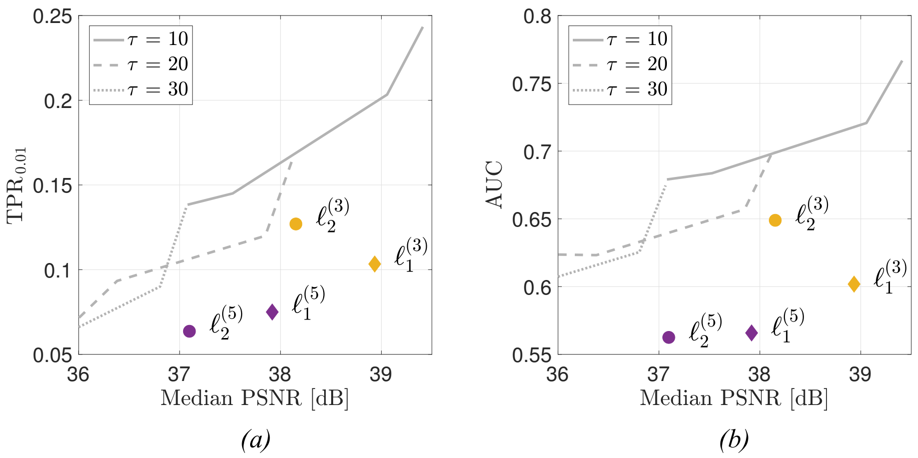


Figure 4.4: *(a) Median* PSNR *vs.* $\text{TPR}_{@0.01}$ *for different inpainting strategies, in comparison with PatchMatch-based (gray lines) patch replacement [92]; (b) Median* PSNR *vs.* AUC *for different inpainting strategies, in comparison with PatchMatch-based (gray lines) patch replacement [92].*

shows encouraging results, both in terms of $\text{TPR}_{0.01}$ and AUC as a function of the median PSNR of inpainted images. As the goal of image anonymization is to reduce $\text{TPR}_{@0.01}$ or approaching AUC $\simeq 0.5$ still granting high PSNR, the best solutions for Figures 4.4*(a)* and 4.4*(b)* are those in the lower right quadrant of the graph. In particular, notice that with the ℓ_1 strategy we are able to reduce $\mathrm{TPR}_{0.01}$ under 11% and to obtain an AUC of 0.6 by still achieving a median PSNR around $39\,\mathrm{dB}$.

As state-of-the-art comparison, Figure 4.4 shows results of the PatchMatch-based image anonymization proposed in [92], adapted to work on color images. This algorithm depends on two parameters: (i) τ is an error threshold; (ii) σ is a smoothing factor. Each gray line represents results obtained for a given τ and changing $\sigma \in \{0.1, 0.75, 2, 4\}$.

This comparison shows that the proposed methodology is an effective alternative to PatchMatch-based anonymization, both in terms of $\mathrm{TPR}_{0.01}$ and AUC. For instance, the ℓ_1 solution for both $B \in \{3, 5\}$ is able to match state-of-the-art performances in terms of median PSNR, while gaining about 0.1 in terms of both $\mathrm{TPR}_{0.01}$ and AUC.

4.3 Image anonymization through CNNs

In this section, we explore the possibilities offered by CNNs in terms of image anonymization. Contrarily to the previous strategy proposed in Section 4.2, we present a method based on the knowledge of the device PRNU, in which an image-wise anonymization loop is built upon a CNN-based noise extractor. Specifically, an autoencoder-inspired fully-convolutional neural network is trained as anonymization function via back-propagation, exploiting the possibilities offered by a recently introduced CNN-based denoising method [115].

It is worth noting that the proposed use of a CNN is different from the typical one. Instead of training a CNN on many images to learn a generalizable method, we "overfit" the proposed CNN on each single image to be anonymized. In other words, we consider the CNN as a parametric operator. We build a loss function to be minimized in order to estimate the CNN parameters. The CNN training is seen as an iterative way of minimizing the CNN loss for each given image.

For the sake of clarity, in the following sections we describe in details the proposed method and the experimental results.

4.3.1 Proposed method

In order to anonymize an image $\mathbf{I}$, we propose an anonymization operator $A\{\mathbf{I}, \mathbf{K}_d\}$ that generates an anonymized version of $\mathbf{I}$, namely $\tilde{\mathbf{I}} = A\{\mathbf{I}, \mathbf{K}_d\}$. As shall be clear from the next section, the design of $A\{\cdot\}$ is such that PSNR between $\mathbf{I}$ and $\tilde{\mathbf{I}}$ is greater than a reference value while the

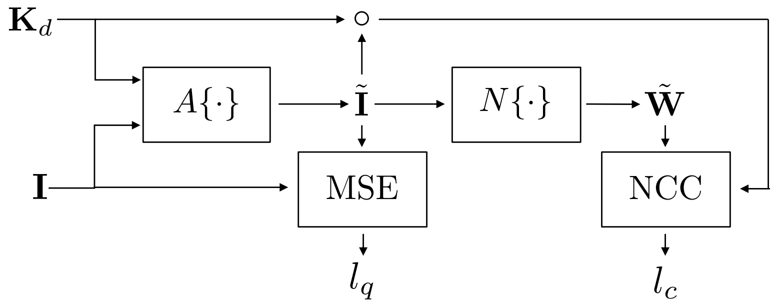


Figure 4.5: *Architecture of the proposed system. An anonymization operator $A\{\cdot\}$ is fed with the input image $\mathbf{I}$ and the relative device PRNU $\mathbf{K}_d$. The anonymized image $\tilde{\mathbf{I}}$ is used to compute a quality loss l_q based on the* MSE *between $\tilde{\mathbf{I}}$ and $\mathbf{I}$. The noise residual $\tilde{\mathbf{W}}$, extracted through a noise extractor $N\{\cdot\}$ from $\tilde{\mathbf{I}}$, is used together with the device PRNU $\mathbf{K}_d$ to determine a correlation loss l_c.*

normalized cross-correlation between the noise residual $\tilde{\mathbf{W}}$ extracted from $\tilde{\mathbf{I}}$ and $\mathbf{K}_d$ is minimized. In an optimal case, it will result that $\mathrm{NCC}(\tilde{\mathbf{W}}, \mathbf{K}_d \circ \tilde{\mathbf{I}}) < \Gamma$, so that it is not possible anymore to bind the anonymized image $\tilde{\mathbf{I}}$ to its camera device with a given confidence γ.

The proposed anonymization method is based on the idea of minimizing a cost function made up of two components:

1. a measure of the difference between input image $\mathbf{I}$ and its anonymized version $\tilde{\mathbf{I}}$;
2. the cross-correlation between the anonymized noise residual $\tilde{\mathbf{W}}$ and the device PRNU $\mathbf{K}_d$.

Figure 4.5 shows the overall working scheme. An image $\mathbf{I}$ and the corresponding device PRNU $\mathbf{K}_d$ are fed as input to the anonymization operator $A\{\cdot\}$. The output of $A\{\cdot\}$ is an anonymized version of $\mathbf{I}$, namely $\tilde{\mathbf{I}}$. The mean square error (MSE) between $\mathbf{I}$ and $\tilde{\mathbf{I}}$ is computed and stored into l_q, the first component of the global cost function. The anonymized image $\tilde{\mathbf{I}}$ is fed as input to the noise extraction operator $N\{\cdot\}$ and the output $\tilde{\mathbf{W}}$ is correlated with the sample-wise product between $\mathbf{K}_d$ and $\tilde{\mathbf{I}}$ to get l_c, the second component of the global cost function. The global cost function l is then defined as $l = (1-\beta) \cdot l_q + \beta \cdot l_c$, where β is a weighting parameter tailored at balancing the trade-off between image quality and anonymization performance.

In the depicted scheme, $N\{\cdot\}$ is a fixed noise extractor, whereas $A\{\cdot\}$ is a denoising operator independently learned on every input pair $(\mathbf{I}, \mathbf{K}_d)$. We require both of them to support gradient computation so that it is possible

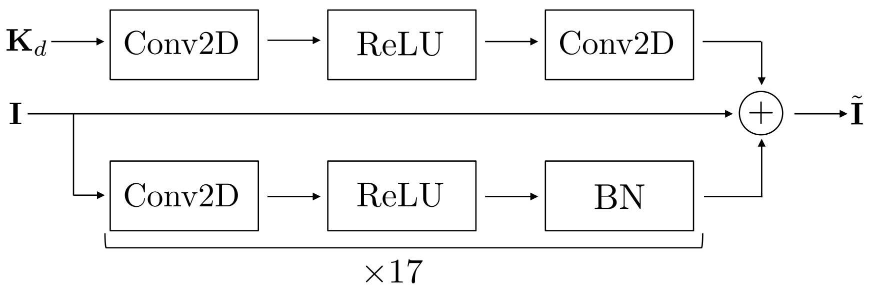


Figure 4.6: *Structure of the proposed CNN-based anonymizer $A\{\cdot\}$. The input image $\mathbf{I}$ is processed through a set of 17 convolutional layers (Conv2D) followed by ReLU non-linearity and Batch Normalization (BN). The reference PRNU $\mathbf{K}_d$ is processed with two convolutional layers separated by a ReLU non-linearity. The output anonymized image $\tilde{\mathbf{I}}$ results from the sample-wise algebraic sum of the input image $\mathbf{I}$ and the two fully-convolutional branches.*

to learn via back-propagation the parameters of $A\{\cdot\}$ as a function of the overall cost function l.

To satisfy the gradient computation capability for $N\{\cdot\}$ we resort to DnCNN [115], a fully-convolutional neural network that shows noise extraction capabilities comparable with the Wavelet-based filtering approach commonly used for PRNU-based image source attribution. DnCNN is composed by a set of 17 convolutional layers composed by 64 filters each with kernel size equal to 3 and padding 1, each followed by ReLU non linearity and batch normalization. The fully-convolutional nature of the network does not require as input a fixed size image and produces as output a noise residual with the same size of the input image.

As for the choice of $A\{\cdot\}$, we exploit an autoencoder structure similar to DnCNN, as depicted in Figure 4.6. The input image $\mathbf{I}$ is processed by a set of 17 convolutional layers (Conv2D), each followed by ReLU non-linearity and batch normalization (BN). The reference PRNU $\mathbf{K}_d$ is fed to a convolutional layer, followed by a ReLU and yet another convolutional layer. The final anonymized image $\tilde{\mathbf{I}}$ results from the sum of the two convolutional processing branches together with the input image itself. The weights of the convolutional layers and the parameters of batch normalization for $A\{\cdot\}$ are learned for every single image via back-propagation, driven by the global cost function l.

The image-wise anonymization process follows as from Algorithm 4.1. An image $\mathbf{I}$, a reference PRNU $\mathbf{K}_d$ and a minimum desired PSNR ($\mathrm{PSNR}_{\mathsf{min}}$) are provided as input. The loss weighting factor β is initialized at 0.1. At every iteration we first compute the anonymized image $\tilde{\mathbf{I}}$,

Algorithm 4.1 Image-wise anonymization process

```
Require: $\mathbf{I}$, $\mathbf{K}_d$, $\mathrm{PSNR}_{\mathrm{min}}$
  $\beta \leftarrow 0.1$
  for $i$ in $\{1, \ldots, 3000\}$ do
    $\tilde{\mathbf{I}} \leftarrow A\{\mathbf{I}, \mathbf{K}_d\}$
    $l_q \leftarrow \mathrm{MSE}(\mathbf{I}, \tilde{\mathbf{I}})$
    $P \leftarrow \mathrm{PSNR}(\mathbf{I}, \tilde{\mathbf{I}})$
    $\tilde{\mathbf{W}} \leftarrow N\{\tilde{\mathbf{I}}\}$
    $l_c \leftarrow |\ \mathrm{NCC}(\tilde{\mathbf{W}}, \mathbf{K}_d \circ \tilde{\mathbf{I}})\ |$
    $l = (1 - \beta) \cdot l_q + \beta \cdot l_c$
    $A\{\cdot\} \leftarrow \textsc{BackPropagate}(A\{\cdot\}, l)$
    if $\textsc{mod}(i, 300) = 0 \wedge P < \mathrm{PSNR}_{\mathrm{min}}$ then
      $\beta \leftarrow \beta/4$
    end if
    if $P > \mathrm{PSNR}_{\mathrm{min}} \wedge l_c < 10^{-4}$ then
      return $\tilde{\mathbf{I}}$
    end if
  end for
  return $\tilde{\mathbf{I}}$
```

together with the MSE loss l_q and the PSNR with respect to the original image $\mathbf{I}$. Then the noise extractor $N\{\cdot\}$ is used to extract a noise residual $\tilde{\mathbf{W}}$ from $\tilde{\mathbf{I}}$ and compute the cross-correlation loss l_c. The global loss l is computed according to the weighting factor β. As all operations in $A\{\cdot\}$ are differentiable, it is possible to back-propagate the error and modify the anonymizer parameters to minimize loss using any iterative optimization algorithm (e.g., stochastic gradient descent in our implementation).

It is not required for $N\{\cdot\}$ and $A\{\cdot\}$ to have a similar structure, as long as both are differentiable operators. Once every 300 iterations, if the PSNR value is smaller than the desired minimum value $\mathrm{PSNR}_{\mathrm{min}}$, the weighting factor β is reduced by a factor 4, in order to raise the importance of the MSE loss l_q over the cross-correlation loss l_c. If the current PSNR is greater than the desired minimum and the cross-correlation loss is small enough (i.e., $l_c < 10^{-4}$), the current anonymized image $\tilde{\mathbf{I}}$ is returned and the optimization stops. At most 3000 iterations of the algorithm are performed, in order to bound the required anonymization time if the early stop condition is not met. A sample of the evolution of $\tilde{\mathbf{I}}$, NCC and PSNR over the iteration is provided in Figure 4.7.

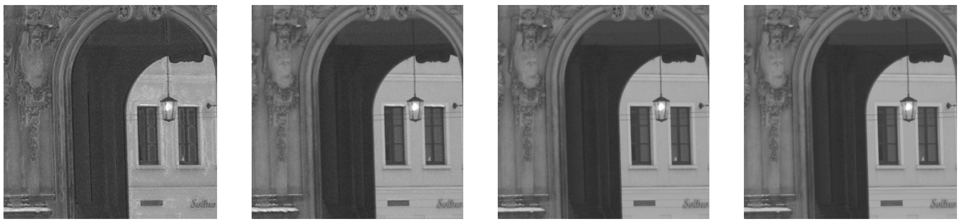


Figure 4.7: *Iterations of the proposed algorithm on a sample image. From left to right the evolution of* $\tilde{\mathbf{I}}$ *at* $i = \{30, 410, 970\}$ *with* NCC *decreasing and* PSNR *increasing. The rightmost picture is the original image* $\mathbf{I}$.

4.3.2 Experiments and results

To state the effectiveness of the proposed approach, we resort to the same dataset and metrics used in state-of-the-art [92] and in Section 4.2 [18]. The dataset is composed of 600 raw natural images, demosaicked with Adobe Lightroom, randomly selected from 6 cameras (Nikon D70, Nikon D70s, Nikon D200, two devices each) from the Dresden Image Database [42]. All the images are cropped in their center to a fixed size of 512×512 pixels. We evaluate the anonymization performance by using two different noise extraction functions:

1. the DnCNN function used as noise extractor within the anonymization scheme, denoted as N_{dn};

2. the Wavelet-based noise exaction function [114] commonly used in PRNU-based works, denoted as N_{wl}.

As for the use of DnCNN as noise extractor, we resort to the pre-trained model available from [115]. We resort to Pytorch [116] as Deep Learning and CNN framework.

The reference PRNU $\mathbf{K}_d$ for each device is estimated from 25 raw flat-field images from the same database, according to N_{wl} as from [20]. All the 600 images are anonymized by varying the $\mathrm{PSNR}_{\mathsf{min}}$ parameter in the set of values $\{37, 38, 39, 40, 41\}$. Each anonymized image is stored as an uncompressed PNG file, thus being quantized to 8-bit as in real case scenario. For each value of $\mathrm{PSNR}_{\mathsf{min}}$ we observe the distribution of the obtained PSNR values. Noise residuals are extracted with N_{dn} and N_{wl} for each anonymized image and than correlated with the 6 camera PRNUs. For each $\mathrm{PSNR}_{\mathsf{min}}$ we compute a receiver operating characteristic curve (ROC)

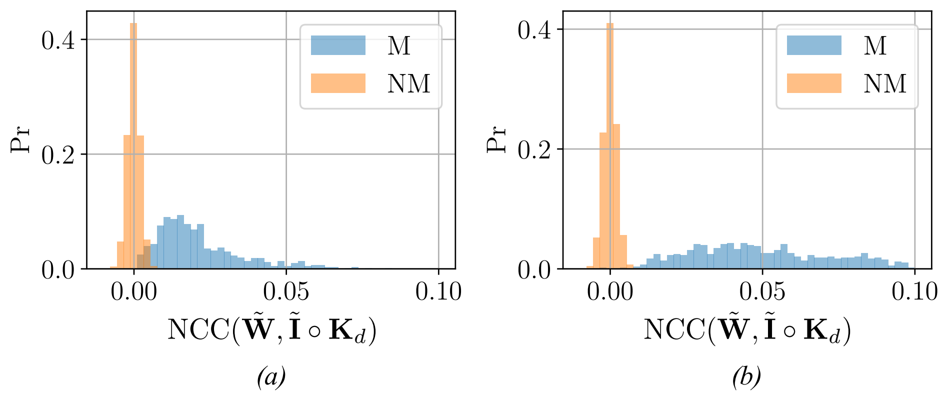


Figure 4.8: *Distribution of normalized cross-correlation values on pristine images using DnCNN (a) and Wavelet (b) noise extractors, for matching image-PRNU pairs* (M) *and non-matching pairs* (NM).

by varying the value of Γ, the threshold used in the cross-correlation test to detect whether an image as being shot from a specific device. From each ROC we extract both the true positive rate value at a false alarm probability $\gamma = 0.01$ ($\mathrm{TPR}_{@0.01}$) and the area under the curve (AUC). Small AUC values indicate good anonymization performance. Small $\mathrm{TPR}_{@0.01}$ values indicate that when accepting a small false-alarm probability it is not possible to bind the picture to its camera device.

Validation of Denoising Operator

First, we need to asses whether DnCNN (N_{dn}) can be used as a reasonable approximation for the widespread Wavelet (N_{wl}) noise extractor tailored to PRNU matching and camera device identification. Figure 4.8 shows the distribution of normalized cross-correlation values (NCC) when N_{dn} and N_{wl} are used as noise extractors from pristine images. In both cases the reference PRNU ($\mathbf{K}_d$) is computed with the Wavelet filter. We can notice that for both noise extractors the discriminability between matching (M) and non-matching (NM) image-camera pairs is preserved, with a slight superimposition of the two distribution for DnCNN.

Figure 4.9 shows the difference in terms of ROC between N_{dn} and N_{wl} on pristine images. The values of AUC reported in the legend show how DnCNN detection performance are slightly lower than the ones of Wavelet, but still above 0.99. This test confirms that DnCNN is able to extract PRNU-based residual information from images, thus justifying its use within our anonymization pipeline.

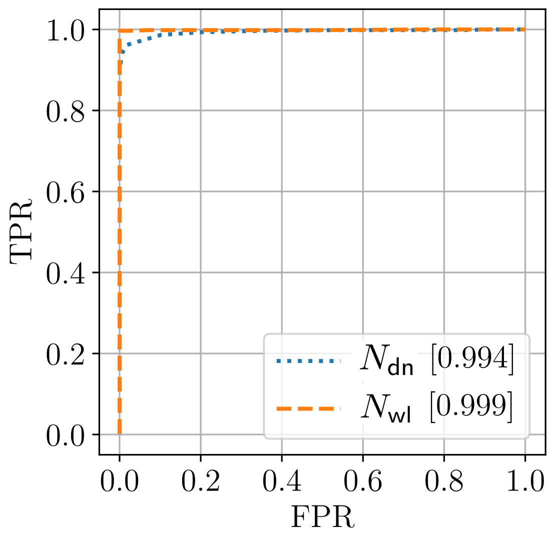


Figure 4.9: *Comparison between* N_{wl} *and* N_{dn} *as noise residual extractors in terms of ROC curves. The* AUC *reported between squared brackets shows almost equivalent performance in terms of detection.*

Minimum PSNR Requirement

As the proposed algorithm uses the $\mathrm{PSNR}_{\mathsf{min}}$ as a driving criteria for the minimum accepted image quality, we are interested in checking whether this criteria is actually met in an experimental fashion. In facts, it might happen that the anonymization loop reaches the maximum number of iterations but the PSNR between $\mathbf{I}$ and $\tilde{\mathbf{I}}$ is still smaller than $\mathrm{PSNR}_{\mathsf{min}}$. Figure 4.10 reports the histograms of PSNR values obtained for various values of $\mathrm{PSNR}_{\mathsf{min}}$. It is possible to notice that for every choice of $\mathrm{PSNR}_{\mathsf{min}}$ the actual values of PSNR are always greater or equal to the minimum bound. This confirms that the proposed iterative method is able to reach convergence in terms of the imposed minimum PSNR requirement.

Image Anonymization

When it comes to verify the effectiveness of the proposed pipeline in reducing PRNU-based device identification, we first compute the distribution of matching and non-matching normalized cross-correlation (ρ) values obtained from anonymized images with noise residuals extracted with DnCNN (N_{dn}) and Wavelet (N_{wl}). Figure 4.11*(a)* shows how the distributions of matching and non-matching ρ values, obtained when noise residu-

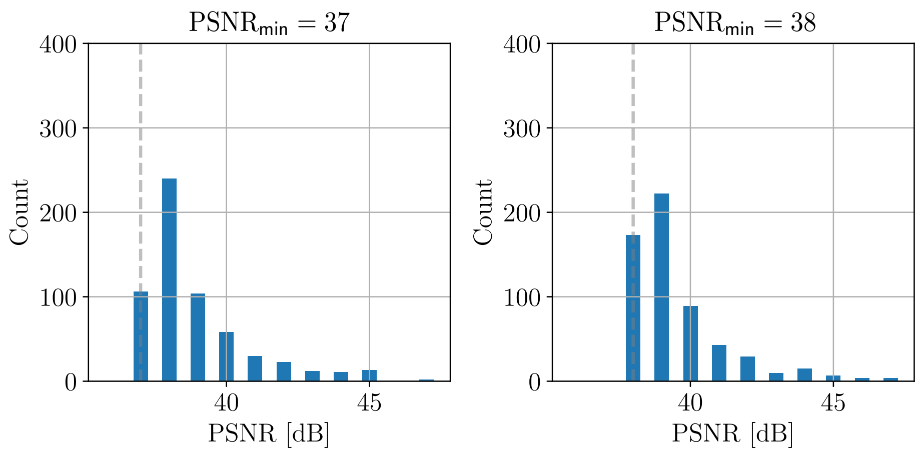


Figure 4.10: *Real* PSNR *distribution when varying* $\mathrm{PSNR}_{\mathsf{min}} \in \{37, 38\}$. *The real* PSNR *values are always greater or equal the the minimum value (vertical dashed gray line). The same behavior is obtained for different* $\mathrm{PSNR}_{\mathsf{min}}$ *values.*

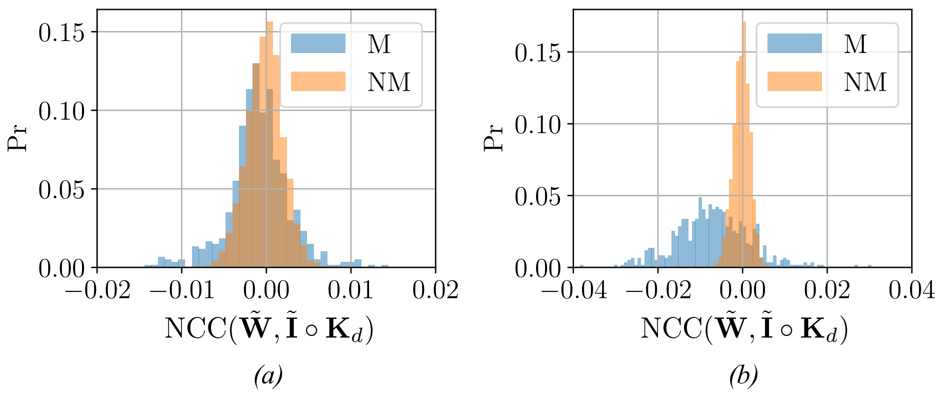


Figure 4.11: *Distribution of normalized cross-correlation values on anonymized images using DnCNN (a) and Wavelet (b) noise extractors, for matching image-PRNU pairs* (M) *and non-matching* (NM) *pairs.*

als are extracted from $\tilde{\mathbf{I}}$ through N_{dn}, are superimposed. This makes practically impossible to bind an anonymized images to the device it comes from. This means that the proposed anonymization pipeline is working in the proper way, thus it has minimized the cross-correlation between the reference PRNU $\mathbf{K}$ and the noise residual extracted through N_{dn}. As we wish to evaluate the effect of the proposed method when the Wavelet-based noise extractor is used on $\tilde{\mathbf{I}}$, Figure 4.11*(b)* shows the distribution

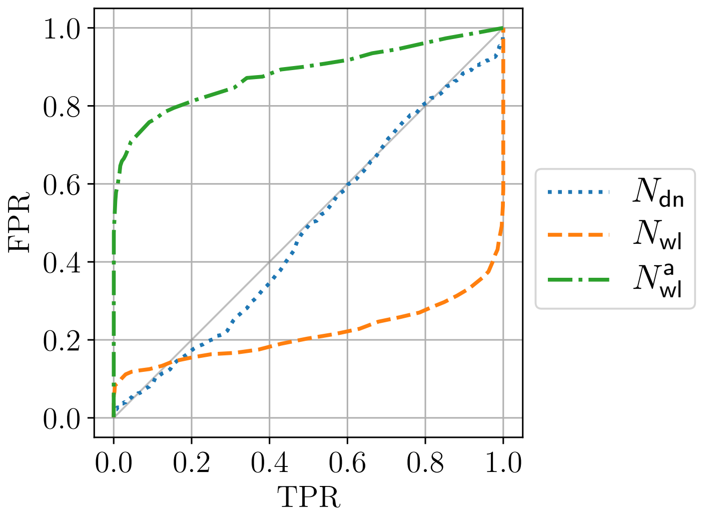


Figure 4.12: *ROC curve after image anonymization process for* $\mathrm{PSNR}_{\mathsf{min}} = 40$.

of matching and non-matching ρ values obtained when noise residuals are extracted with N_{wl}. We can immediately spot two differences with respect to the N_{dn} extractor: i) the mean of the matching values is not anymore zero, but it is shifted toward negative values; ii) the variance of matching cross-correlations is way higher than the variance of non-matching cross-correlations. A forensic investigator acting in a blind way, without the knowledge of the proposed anonymization pipeline, might use the cross-correlation test to asses whether an image $\tilde{\mathbf{I}}$ under investigation comes from a camera whose PRNU is $\mathbf{K}$. However, a smart investigator would also perform another test, evaluating the absolute value of the normalized cross-correlation, thus building a symmetric test $|\rho(\mathbf{W}, \mathbf{K}_d \circ \mathbf{I})| > \tau$. In the plots, we refer to the results obtained with the standard Wavelet detector with N_{wl}, while the results obtained with the Wavelet symmetric detector are denoted as $N^{\mathsf{a}}_{\mathsf{wl}}$.

Figure 4.12 shows the ROC on anonymization detection for $\mathrm{PSNR}_{\mathsf{min}} = 40$. If N_{dn} is used to extract the noise residual from $\tilde{\mathbf{I}}$ we get almost perfect anonymization performance. This confirms that the anonymization loop, based on the minimization of the cross-correlation value between $\mathbf{K}_d \circ \tilde{\mathbf{I}}$ and $\tilde{\mathbf{W}}$ extracted through N_{dn}, is effectively working as expected. When noise residuals are extracted from $\tilde{\mathbf{I}}$ through the Wavelet-based function and the unidirectional test is used (N_{wl}), the detection performance are severely affected. However, resorting to the symmetric detector ($N^{\mathsf{a}}_{\mathsf{wl}}$) shows that in

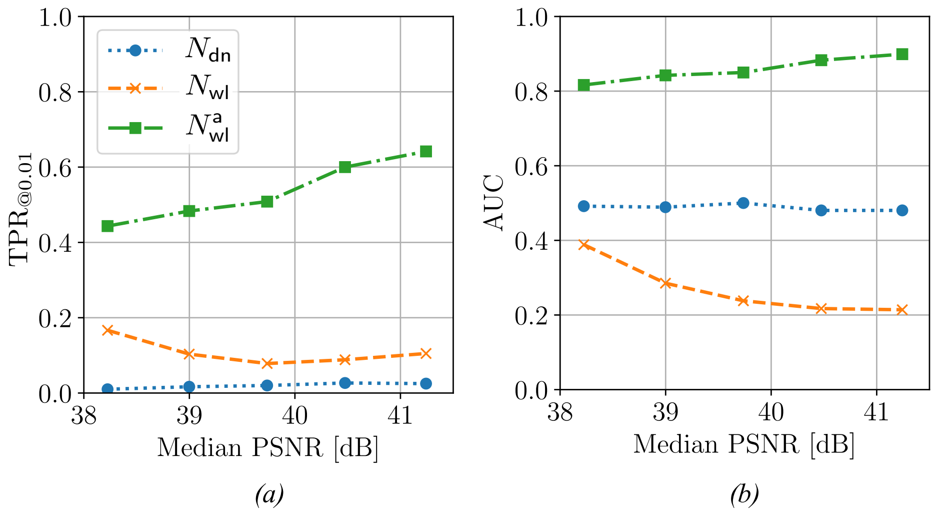


Figure 4.13: $\mathrm{TPR}_{@0.01}$ *(a) and* AUC *(b) when varying* $\mathrm{PSNR}_{\mathsf{min}}$.

fact the detection performances are affected, but are not as bad as when the asymmetrical detector is used.

A final result is shown in Figure 4.13, where two standard metrics in anonymization are presented. Figure 4.13 respectively reports $\mathrm{TPR}_{@0.01}$ and AUC for several median PSNR values. Each point is obtained by setting $\mathrm{PSNR}_{\mathsf{min}}$ to $\{37, 38, 39, 40, 41\}$. The almost zero $\mathrm{TPR}_{@0.01}$value for N_{dn} and the almost constant 0.5 value for AUC are assessing that the anonymization cycle is working properly if the noise extraction function used in the anonymization loop is the same as the one used for analysis purposes. When a different noise extraction function is used and a forensics investigator is aware of the attack ($N^{\mathsf{a}}_{\mathsf{wl}}$) the anonymization is not guaranteed anymore.

4.4 Conclusions

In this chapter, we propose two strategies for anonymizing images against PRNU-based detectors.

The first approach leverages image inpainting to reconstruct image pixels and edge processing to increase image visual quality. We test different inpainting strategies, showing that it is possible to attenuate PRNU traces even exploiting simple inpainting solutions. Considering that results are competitive with state-of-the-art blind PRNU removal solutions [92], the investigated pipeline proves an interesting alternative method for image anonymization. Moreover, the proposed framework is computationally efficient because of its high parallelization potential. Indeed, inpainting is computed in parallel on different versions of the image, and results are merged at the end.

The second approach works in a scenario in which the specific PRNU to be removed is assumed to be known. Despite the previous approach and state-of-the-art methods achieve better anonymization performance, we believe this strategy shows a different perspective on the topic, as the proposed solution makes use of a CNN in an uncommon fashion. Indeed, the CNN is seen as a parametric operator. The training phase is used to estimate CNN parameters by minimizing a loss function on a single image. Given these premises, the proposed method works by overfitting a specific CNN to each input image.

From the adversarial forensic point-of-view, results show an interesting aspect. If the denoising operators used for PRNU testing and within the anonymization network match (i.e., DnCNN is used), images are strongly anonymized. If the analyst makes use of a different denoising operator for PRNU testing (i.e., the Wavelet-based one), anonymization may be effective or not depending on the used correlation test. In reality, denoising operator matching is not needed by an attacker, given that the analyst is not informed about the possibility of an attack. If analysts know about possible attacks, they can use the symmetric test to avoid being completely fooled.

CHAPTER 5

Geophysical applications

The know-how acquired over these three years on regularization techniques for inpainting and denoising of digital images, in addition to the matured experience on convolutional neural networks, has generated the possibility to solve similar problems belonging to completely different fields of research. For instance, even if including higher dimension data, different constraints and priors, denoising and inpainting problems are commonplace in geophysics applications as well. Some examples can be found in removal of random and/or coherent noise in seismic pre-processing workflow, or in reconstruction of regular and densely-sampled seismic traces. Indeed, there is a keen interest in the geophysics community regarding data denoising, interpolation and extrapolation.

Given these premises, in this chapter we present some results achieved by exploiting CNNs for the goal of denoising and/or interpolation of 2D common shot gathers. In particular, inspired by the great contributions achieved in image processing and computer vision, we investigate a particular architecture referred to as *U-net*, which implements a convolutional autoencoder able to describe the complex features of clean and regularly sampled data for reconstructing the corrupted ones. In the following, we first introduce the problem and the state-of-the-art, then we describe the details of the proposed method.

5.1 Seismic data interpolation and denoising

Seismic processing and imaging methods are essential to discover, localize and characterize economically worthwhile geological reservoirs, such as hydrocarbons accumulations, and to manage the extraction of the resources stored in them. However, since easy-to exploit resources are dramatically reducing and exploration targets are more and more complex, the requirements for the quality of seismic data, both in term of signal-to-noise ratio and of regularity and density of its sampling, are constantly increasing.

Unfortunately, various types of random and coherent noise, depending on the environment and on the acquisition technology, often corrupt seismic data sets. An additional problem is that economic limitations, cable feathering in marine case, environmental constraints and elimination of badly acquired traces cause irregular spatial sampling in almost all seismic acquisitions.

Most state-of-the-art seismic processing algorithms, such as reverse-time-migration [117], full-waveform-inversion [118] and surface related multiple elimination [119] benefit from high quality regularly sampled data. Consequently, the vast majority of seismic processing workflows require data pre-processing steps, including effective denoising and trace interpolation algorithms. Moreover, due to the increasing size of the acquired data, a key factor of these procedures for industrial application is their computational burden, in terms of both memory requirements and computational time. Given these premises, the following lines summarize the state-of-the-art for the problem of seismic data interpolation and/or denoising.

5.1.1 State-of-the-art in geophysics

The problems of trace interpolation and noise attenuation have been widely investigated, either simultaneously or separately. Among the dozens of interpolation methods proposed so far, we can roughly identify four main categories.

Model-based algorithms implement an implicit migration-demigration pair [120, 121]. A major drawback of these techniques is that their performance is strongly affected in case of complex structural burden.

A second approach, for both denoising and interpolation, is based on prediction filters [122–125], which assume seismic data to be a (local) superposition of plane w1aves. However, these methods target regularly sampled data, which is a heavy limitation.

Due to their repetitive features, clean seismic data are intrinsically low-rank in the time-space domain. Conversely, noise and missing traces increase the rank of the data [126]. Therefore, algorithms recasting the interpolation (and denoising) problems as rank reduction and matrix/tensor completion have been largely studied in the past decade as third alternative to the problem [127–131].

A great amount of denoising and interpolation algorithms exploit a transform domain where the clean signal can be represented only by few non-zero coefficients and therefore clean data and noise are more easily separable. The rationale behind this forth family of methods is that noise and missing traces map in non-sparse artifacts in the transform domain.

Several fixed-basis sparsity-promoting transforms have been widely used also for seismic data interpolation. Among the various approaches, coming from different fields, we can cite: the Fourier transform [132], the Hilbert-Huang transform [133], the time-frequency peak filtering [134], the Radon transform [135], different curvelet-like transforms [136–140] and the EMD-seislet transform [141].

These methods assume the ability to describe the data in terms of a linear combination of atoms taken from a dictionary. The aforementioned transform methods implicitly assume regularity of the data described by analytic models, resulting in predefined fixed dictionaries. However, these dictionaries can be thought as defining only a subset of the transforms methods.

Alternatively, data driven sparse dictionaries can be learned directly from the dataset. In other words, these methods assume that clean signals are a linear combination under a sparsity constraint of the atoms in a learned overcomplete dictionary. Learned dictionaries, in the form of explicit matrices for small patches, usually better match the complex data characteristics. For instance, denoising results obtained using double sparse dictionary learning and outperforming fixed dictionary transforms have been reported in [142] and [143], combining the dictionary learning based sparse transform with the fixed-basis transform, which is called double-sparsity dictionary. Recently, [144] introduced a joint seismic data denoising and interpolation method using a masking strategy in the sparse representation of the dictionary.

5.1.2 A novel method based on CNNs

In recent years, outstanding advancements brought by deep learning and CNNs have greatly impacted whole signal and image processing community. In this context, innovative strategies for data interpolation and denoising based on deep learning have been proposed in many image processing tasks. Indeed, solutions based on CNNs are nowadays often exceeding state-of-the-art results. However, these methods have barely started to be explored by the geophysical community for the problems of denoising and interpolation. Promising results for the aforementioned tasks have been reported through residual neural networks [145, 146], generative adversarial networks [147] and convolutional autoencoders [21, 148].

In the following lines, we present our method to interpolate and denoise corrupted seismic data in the shot-gather domain. In particular, inspired by the important contributions achieved in image processing problems, we exploit a properly trained *U-net* [149] as a strongly competitive strategy for noise attenuation and reconstruction of missing traces in pre-stack seismic gathers. We provide examples on synthetic and field data showing promising performances on either denoising, interpolation, or joint denoising and interpolation problems.

5.2 Seismic data interpolation and denoising through CNNs

In this section, we first report details about the formulation of the tackled problems, namely interpolation and/or denoising of seismic data. Then, we provide some background concepts on Convolutional Autoencoders (CAs) and how to exploit them for our specific goals.

5.2.1 Problem formulation

We focus on the problem of reconstructing seismic gathers which have been corrupted by irregular trace sampling and/or additional noise. Formally, we represent each original non-corrupted seismic gather as $\mathbf{I}$ and its corrupted version is denoted as $\bar{\mathbf{I}}$. Our goal is to estimate a clean and dense version of the seismic data, namely $\hat{\mathbf{I}}$, as similar as possible to the original corresponding gather $\mathbf{I}$.

In order to solve this problem, we make use of a particular kind of convolutional neural network named convolutional autoencoder (CA). Our choice is motivated by the great capability of CA in learning compact representations of the data, and by the strong computational efficiency in reconstructing the corrupted ones. In the following, we report some backgrounds on CAs, introducing the specific network architecture exploited for the prescribed task.

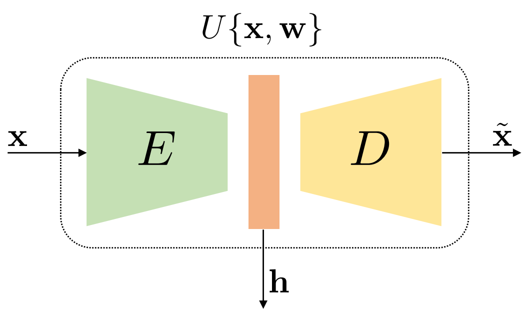


Figure 5.1: *Scheme of a Convolutional Autoencoder architecture.*

5.2.2 Convolutional autoencoders for interpolation and denoising

Convolutional autoencoders (CAs) are convolutional neural networks whose architecture can be logically split into two separate components: the encoder and the decoder. The CA structure is sketched in Figure 5.1:

(i) the encoder, represented by the operator $E\{\cdot\}$, maps the input $\mathbf{x}$ into the so called hidden (or latent) representation $\mathbf{h} = E\{\mathbf{x}\}$, which is the innermost encoding layer of the autoencoder, compressing the input $\mathbf{x}$ into a high-level representation [150];

(ii) the decoder, represented by the operator $D\{\cdot\}$, transforms the hidden representation into an estimate of the input $\tilde{\mathbf{x}} = D\{\mathbf{h}\}$.

For image processing problems, CA proves to be a very powerful instrument for inpainting and denoising tasks [151, 152]. The rationale behind the use of CA for inpainting and denoising shares some common concepts with the transform-based and dictionary learning techniques. Indeed, CA is trained so that the encoder part results in a compact representation of clean data, where the interference due to noise and missing samples is not mapped. Therefore, if the compact representation is correctly built, the result of the decoder is a *dense* clean image without missing samples. Consequently, it is possible to train a CA to learn a hidden representation of the clean data in common shot gathers and then to recover clean and densely sampled gathers from noisy and scattered ones.

In particular, in this work we exploit a CNN architecture known as *U-net*. Originally designed for image segmentation problems and then used for several different tasks [153], the *U-net* is named after the shape it is usually graphically represented with. Indeed, *U-net* shares a large part of the architecture with classical CA. However, in a *U-net*, the representations of the input obtained at different levels of the encoder are directly concatenated to the corresponding decoder levels. This peculiarity typically allows to preserve the structural integrity of the image and to enable a very precise reconstruction. For the sake of brevity, we refer to [149] for a detailed explanation of these architectures.

Continuing the analogy with the transform-based methods, we can think the trained *U-net* as an instrument implicitly providing a multi-scale/multi-resolution compact representation, able to describe the complex features of clean seismic data where noise and missing data are not modeled. We can think at the interpolation and denoising task as an image transfer problem, with the goal of transforming gathers corrupted by noise and/or missing traces into regularly sampled clean gathers.

5.2.3 Reconstructing the Corrupted Gathers

In this section we report the technical details concerning the used network architecture and then we describe two strategies for network training, to be used according to the specific data corruption. Finally, we describe the system deployment, that is the procedure designed to process, after the training stage, the actual gather we want to reconstruct.

Implementation of U-net

In order to focus on local portions of the gathers and to ensure a sufficiently large amount of data under analysis, we work in a patch-wise fashion. Specifically, we divide each shot gather into K patches of size $N \times N$. Then, we always consider a single patch as input to the network. Notice that, for what regards the *U-net* implementation, patches can be extracted from the gather with or without overlap, being the *U-net* architecture independent from the patch selection strategy.

The CNN we propose takes inspiration by our previous work regarding the only interpolation task [21], but introduces a few novelties in order to simplify the network architecture at highest levels and enhance the system efficiency without drops in performances. For instance, as seismic data typically has a value range very different from that of natural digital images for which CNNs are typically studied, each patch is multiplied by a constant gain G (i.e., $G = 2000$ in the following experiments), which proves to be an effective data normalization procedure in terms of convergence speed and achieved validation results. Moreover, we do not need to consider the gradient computation and the batch normalization after the first convolutional layer, thus making the proposed approach leaner.

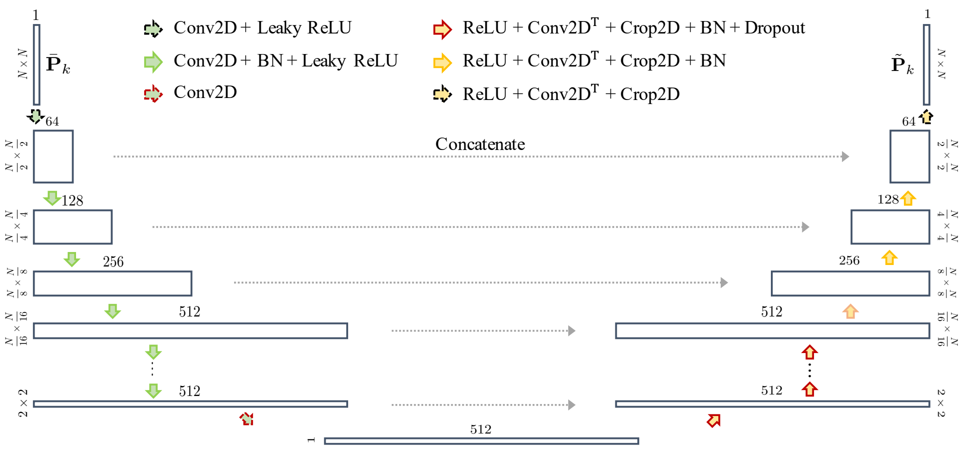

Conv2D + Leaky ReLU
Conv2D + BN + Leaky ReLU
Conv2D
ReLU + Conv2D^T + Crop2D + BN + Dropout
ReLU + Conv2D^T + Crop2D + BN
ReLU + Conv2D^T + Crop2D
Concatenate
$\bar{\mathbf{P}}_k$
$\tilde{\mathbf{P}}_k$
$N \times N$
$\frac{N}{2} \times \frac{N}{2}$
$\frac{N}{4} \times \frac{N}{4}$
$\frac{N}{8} \times \frac{N}{8}$
$\frac{N}{16} \times \frac{N}{16}$
2×2
1
64
128
256
512
512
512

Figure 5.2: *Architecture of the used* U-net.

Considering that corrupted gathers are labeled as $\bar{\mathbf{I}}$, the generic k-th corrupted patch given as input to the network is denoted as $\bar{\mathbf{P}}_k$.

As *U-net* like architectures turn out to be the state-of-the-art for the tasks of image inpainting [154, 155] and denoising of medical images [156], we follow the trend started by [149], exploiting a *U-net* architecture composed by the blocks shown in Figure 5.2:

1. A number of stages where a 2D Convolution with filter size 4×4 and stride 2×2, sometimes followed by Batch Normalization (BN) and/or Leaky ReLU, is performed. These stages lead to the hidden representation (i.e., the result of the encoding part). Specifically, we do not include the BN stage in the outermost layer and after the 2D Convolution leading to $\mathbf{h}$. It is worth noting that the number of filters increases from 64 to 512 as we go deep into the network.

2. The same number of stages as before where a ReLU, a 2D Transposed Convolution with filter size 4×4 and stride 2×2 and a 2D cropping, possibly followed by BN and Dropout are performed. In this case, we use BN on all layers except from the last one, while Dropout is used only in the initial layers, until we reach a patch dimension of $\frac{N}{16} \times \frac{N}{16}$. In each stage we concatenate the result of the corresponding encoding stage as in a typical *U-net* fashion. Note that the number of filters is gradually diminished as we go up in the right path of the network (i.e., decoding path). The last stage outputs the patch $\tilde{\mathbf{P}}_k$, of the same size of the input patch.

Additionally, the overall architecture scales according to the patch di-

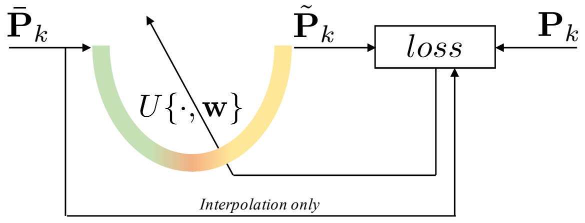


Figure 5.3: *Sketch of the training phase.*

mension N, which can be selected depending on possible application-driven constraints. Anyway, as the network can be characterized by more than 40 million parameters, it needs to be trained on a significant amount of seismic images as any typical deep learning solutions.

5.2.4 U-net training

Once defined the network architecture, the key point is the design of the training strategy through a proper definition of the cost function tailored to our specific problem. Indeed, the *U-net* defines a parametric model $\tilde{\mathbf{x}} = U\{\mathbf{x}, \mathbf{w}\}$, between the output $\tilde{\mathbf{x}}$ and the input $\mathbf{x}$ and network weights $\mathbf{w}$. The training phase consists in estimating the network weights $\mathbf{w}$ through the minimization of a distance metrics between the network input and its output. This distance is usually referred as *loss* function, and its minimization is carried out using iterative techniques (e.g., stochastic gradient methods, etc.).

Specifically, as shown in Figure 5.3, we train the network in order to transform patches $\bar{\mathbf{P}}_k$, extracted from gathers corrupted by noise and/or missing traces, into regularly sampled and clean patches $\tilde{\mathbf{P}}_k$. As in any supervised learning problem, we assume to have a training dataset $\mathcal{D}_T$ and a validation dataset $\mathcal{D}_V$, each one composed by pairs of corrupted/uncorrupted gathers $(\bar{\mathbf{I}}, \mathbf{I})$.

These datasets are exploited for estimating the network parameters $\mathbf{w}$ and to decide when to stop the cost function minimization process. Actually, the training stage slightly differs depending of the specific problem: denoising only, interpolation only, and joint denoising/interpolation.

For denoising only and joint denoising/interpolation, model weights are estimated by minimizing the *loss* function between $\mathbf{P}_k$ and $\tilde{\mathbf{P}}_k$, defined as the squared error over all patches belonging to gathers in the training set

$\mathcal{D}_T$. Formally,

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{\mathbf{P}_k \in \mathcal{D}_T} \left\| \mathbf{P}_k - \tilde{\mathbf{P}}_k(\mathbf{w}) \right\|_{\mathrm{F}}^2 , \tag{5.1}$$

where $\|\cdot\|_{\mathrm{F}}$ represents the Frobenius norm and, with a slight abuse of notation, $\mathbf{P}_k \in \mathcal{D}_T$ denotes patches extracted from gathers belonging to training dataset.

However, for the task of interpolation only, we a-priori know that only some samples need to be reconstructed by the network (i.e., the missing ones), whereas the others can be left untouched. For this reason, the *loss* is evaluated only on reconstructed samples. This is implicitly performed by adding a masking stage that sets to zero all uncorrupted traces. Network weights are then estimated as

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{\mathbf{P}_k \in \mathcal{D}_T} \left\| (\mathbf{P}_k - \tilde{\mathbf{P}}_k(\mathbf{w})) \circ \mathbf{M} \right\|_{\mathrm{F}}^2 , \tag{5.2}$$

where $\mathbf{M}$ is a binary mask of size $N \times N$ defined as

$$[\mathbf{M}]_{i,j} = \begin{cases} 1 & \text{if } \left[\bar{\mathbf{P}}_k\right]_{i,j} \text{ is a missing sample} \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

As in standard neural network training, we follow an iterative procedure to minimize either (5.1) or (5.2), stopping at the iteration where the mean squared error over all the patches $\mathbf{P}_k \in \mathcal{D}_V$ is minimum.

Specifically, we use Adam optimization algorithm [48], with learning rate and patience (i.e., the number of epochs with no improvement after which training will be stopped) initialized at 0.01 and 10, respectively. The former is decimated while the latter is halved in presence of plateau of the cost function. In general, we train the network for a maximum number of 100 epochs, although we verified the smallest loss on validation patches is often achieved within the first 30 training epochs.

System Deployment

When a new corrupted gather $\bar{\mathbf{I}}$ belonging to evaluation set $\mathcal{D}_E$ is under analysis, its recovered version is estimated following the scheme depicted in Figure 5.4.

First of all, a set of K patches is extracted from image $\bar{\mathbf{I}}$ as described in the previous Section. Then, each patch $\bar{\mathbf{P}}_k$ is processed by the *U-net* architecture in order to estimate the patch $\tilde{\mathbf{P}}_k$.

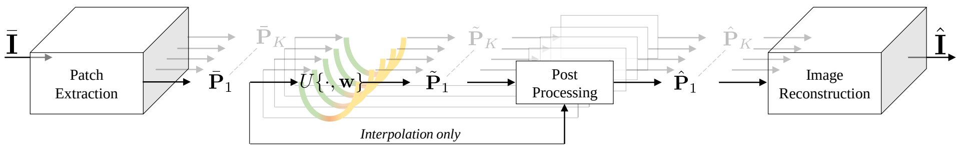


Figure 5.4: *Sketch of the proposed procedure for recovering each corrupted gather* $\bar{\mathbf{I}}$.

Following the same logic of the training phase, the estimated patches are post-processed in slightly different ways according to the specific goal. For denoising only and joint denoising/interpolation, each patch $\tilde{\mathbf{P}}_k$ simply undergoes a denormalization step, thus it is divided by the gain G to obtain the output patch $\hat{\mathbf{P}}_k$. Concerning the task of interpolation only, it is reasonable to leave the known samples untouched in the final estimated patch. Therefore, exploiting the binary mask defined in (5.3), each patch $\hat{\mathbf{P}}_k$ is obtained as

$$\hat{\mathbf{P}}_k = \frac{\bar{\mathbf{P}}_k \circ \bar{\mathbf{M}} + \tilde{\mathbf{P}}_k \circ \mathbf{M}}{G}, \tag{5.4}$$

being $\bar{\mathbf{M}}$ the the logical complement (i.e., the negation) of $\mathbf{M}$.

Eventually, in order to reconstruct the image gather $\hat{\mathbf{I}}$, all the estimated patches $\hat{\mathbf{P}}_k$ are re-assembled together, sample-wise averaging the overlapping portions if some overlap between patches was used during patch extraction procedure.

5.3 Experimental results

In this section we present the result of our experiments, obtained on well-known public datasets. Specifically, we evaluate our methodology over both synthetic and field data. First, we introduce the accuracy metrics we exploit for evaluating the results, and we show a tutorial example considering a very simple case of study. Second, we separately validate the performances of the proposed interpolation and denoising strategies on both synthetic and field data. Finally, we investigate the combined interpolation and denoising problem also comparing our method against a recent solution.

5.3.1 Accuracy metrics

We evaluate the performances of our method in reconstructing each entire seismic image belonging to the evaluation set $\mathcal{D}_E$, namely the corrupted gathers which have never been seen by the *U-net*. The accuracy metrics is the same used by [146] and [147], i.e., the SNR, defined as the ratio between the power of the original gather and the power of the reconstruction error:

$$\mathrm{SNR} = 10\,\log_{10}\frac{||\,\mathbf{I}\,||_{\mathrm{F}}^{2}}{||\,\mathbf{I}-\hat{\mathbf{I}}\,||_{\mathrm{F}}^{2}} \quad \mathbf{I}\in\mathcal{D}_{E}. \tag{5.5}$$

5.3.2 Problem 1: Interpolation of missing traces

In this section we present the network performances in interpolating gathers with missing traces. In particular, we start showing results achieved over a synthetic dataset, considering various interpolation situations. Finally, we evaluate the proposed strategy on real field data.

Synthetic data set

The reference dataset used to systematically explore the results is extracted from the well known synthetic BP-2004 benchmark [157]. In particular, we work with 1348 shot gathers, cropped at the first 1152 traces (taking the source as reference) and at the first 1920 time samples/trace. The central frequency of each trace is 27Hz, sampled every $\delta_t = 6\cdot 10^{-3}$seconds, and the group spacing is 12.5m. In order to properly evaluate the proposed method, we randomly split the dataset into training, validation and evaluation, using 250 shot gathers for training and validation (further split on 75% of images for training set $\mathcal{D}_T$, and 25% for validation set $\mathcal{D}_V$), and the remaining for evaluation set $\mathcal{D}_E$.

Interpolation of uniformly distributed missing traces

The first experiment investigates the situation of randomly located missing traces with uniform distribution. This choice follows the main reasoning of the works proposed in literature [21, 132, 144].

In order to simulate a seismic acquisitions with randomly missing traces, we extract 3 different datasets from the reference one, deleting a percentage H of the available data traces. To be precise, for each shot gather $\mathbf{I}$, we randomly delete the $H\%$ of its traces, $H\in\{10, 30, 50\}$, obtaining a holed gather $\bar{\mathbf{I}}$.

As shown in the Section regarding the *U-net* implementation, we work in patch-wise fashion for reconstructing the corrupted gathers. Specifically, each gather entering the network is initially split into a plurality of squared patches, with dimensions $N\times N$. In light of this, we perform an initial experiment to analyze the behaviour of network output as a function of the

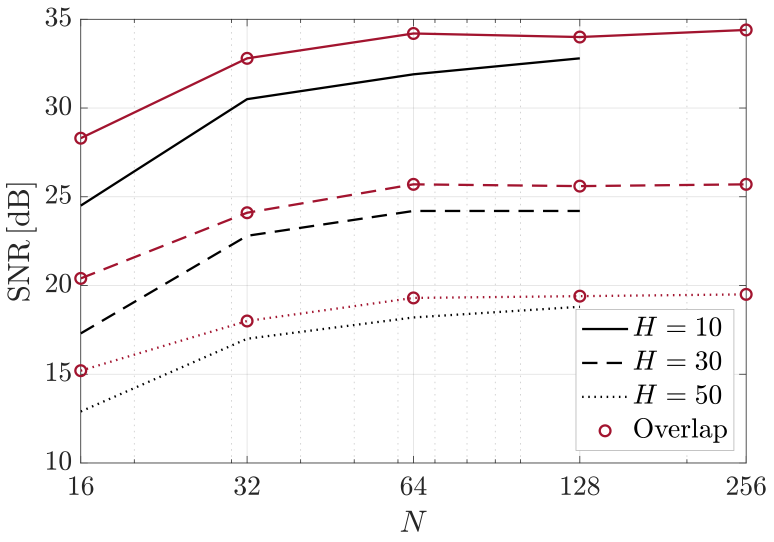


Figure 5.5: *Average* SNR *[dB] achieved on gathers belonging to $\mathcal{D}_E$, as a function of patch dimension N and overlap.*

specific input data. The goal of this primary investigation is to select a good patch extraction method, that is, the strategy leading to the highest reconstruction accuracy on the evaluation set. We consider different values for N, namely $N \in \{16, 32, 64, 128, 256\}$, and we evaluate the cases of non-overlapping patches and of patches extracted with an overlap of $N/2$ in both directions.

To evaluate the *U-net* performances according to the chosen patch extraction method, we use SNR defined in (5.5). Figure 5.5 shows the average SNR achieved over gathers belonging to evaluation set, with and without the overlap between the extracted patches. Note that the case $N = 256$ does not include results without overlap because the gather dimensions are not integer multiples of this value. It is noticeable that small values of N are not good solutions for reconstructing the corrupted images, probably because the *U-net* needs to analyze more samples together in order to find a significant hidden representation of the input patch. As expected, introducing some overlap during patch extraction always returns better performances than just selecting adjacent patches. This is due to two main factors: first, selecting overlapped patches increases the amount of data seen by the network and reasonably improves its performances; second, in the image reconstruction phase, the overlapping portions of the patches are sample-wise averaged, decreasing the possibility to generate undesired edge/border effects.

Even if selecting an overlap of $N/2$ gives slightly better results, one consideration must be done. At training stage, we found out that a good strategy is to group in a single batch all the patches extracted from the same shot gather, ending up with a batch size (i.e., the amount of *patches* in a batch) strictly dependent on N. Notice that the number of *samples* per batch does not change with N, if patches are not overlapped. Conversely, in case of overlapped patches, the number of samples per batch increases, as some samples belong to multiple patches. Therefore, the higher the overlap, the larger the amount of GPU memory required in training phase. If the absence of overlap requires a GPU memory usage more or less equal to 4GB for every N, in case of overlap the required space increases in a quadratic fashion.

Therefore, considering that the achieved SNR performances of the two methodologies (overlapped and non-overlapped patches) are not so far, we choose the patch extraction strategy which selects only adjacent and non-overlapping patches. For this reason, hereinafter we only investigate the network behavior considering non-overlapping patches, as overlapping ones would make the solution impractical in the majority of cases.

Regarding the patch dimension N, as the SNR curve monotonically increases with the patch dimension but without dropping performances in terms of memory usage, we select $N = 128$ for all the experiments. We end up with batches of 135 non overlapping patches with dimensions 128×128 extracted from each shot gather. The process involves more than $25\,000$ training patches, more than $8\,500$ validation patches, and more than $145\,000$ testing patches for each dataset.

Regarding the results, we are able to achieve SNR of 32.8dB, 24.2dB and 18.8dB for $H = 10, 30$ and 50, respectively. The processed gathers do not visually show any artifacts due to the interpolation method, even with $H = 50$, as depicted by the example in Figure 5.6. For what concerns the required computational time, we need more or less 75 minutes for training and validate the network over 100 epochs. Despite this could seem a quite huge amount of time, once training has been done, we are able to recover a corrupted shot gather belonging to the test set in less than 0.3 seconds. Specifically, we run our tests on a Workstation equipped with Intel Xeon E5-2687W v4 (48 Cores @ 3 GHz), RAM 252 GB and 1 TITAN V (5120 CUDA Cores @ 1455MHz), 12 GB.

Interpolation of bursts of missing traces

Uniform distribution of missing traces, described by the percentage H, allows the evaluation of average reconstruction performances of the in-

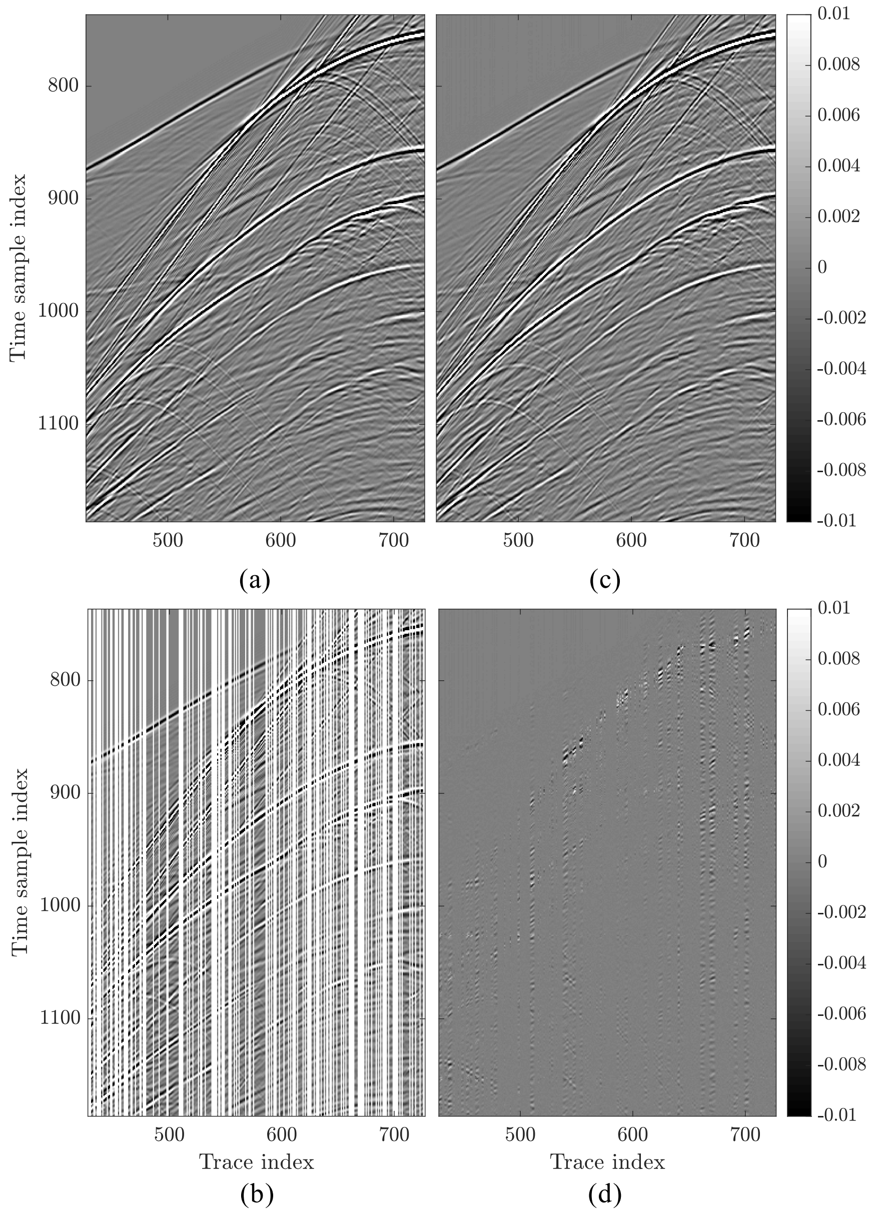


Figure 5.6: *Example of data interpolation considering one gather of the synthetic dataset. (a) depicts the original gather* $\mathbf{I}$*, cropped in its central portion with size* 450×300*; (b) reports the corrupted gather* $\bar{\mathbf{I}}$*, with* 50% *of randomly missing traces, (c) shows the reconstructed gather* $\hat{\mathbf{I}}$*; (d) depicts the reconstruction error, which is the difference between reconstructed and original shot gather.*

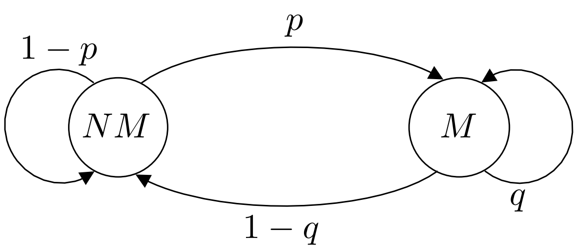


Figure 5.7: *Markov chain of the burst corruption model.*

terpolation. However, in order to have a more detailed description, here we study the performances of the proposed *U-net*-based interpolation on a more sophisticated corruption model. Basing on the consideration that missing traces (due for instance to spatial obstacles) are likely to appear in groups, we propose a bursty missing traces model inspired by the packet loss models of telecommunication networks [158]. The term *burst* is taken from the field of telecommunication networks, and refers to groups of consecutive events (in this case, groups of consecutive missing traces). The more bursty a distribution of missing traces is, the more the missing traces are likely to cluster.

In particular, the model is a two states Markov model described by two parameters, α and β: α refers to the probability of a missing trace, while β is the average length of the burst, i.e., the average number of missing traces which are adjacent one another.

The Markov chain of the model is depicted in Figure 5.7, where NM represents the non-missing trace state whereas M is the state for missing trace. The probability to find a corrupted trace, given that the previous one (in the spatial dimension) was missing, is q, while the probability to pass from a non missing trace to a missing one is p. These probability values can be derived from α and β, formally,

$$q = 1 - \frac{1}{\beta}, \quad p = \frac{\alpha}{\beta(1-\alpha)}. \tag{5.6}$$

Exploiting this model, we can simulate more realistic scenarios, where clusters of adjacent missing traces can occur, due for instance to environmental constraints or sudden interruptions during acquisitions. In order to test our method on this missing trace distribution, we select various percentage of missing traces $\alpha \in \{10, 30, 50\}\%$ with average burst length $\beta \in \{1, 2, 3\}$, corresponding to $12.5\text{m}, 25\text{m}$ and 37.5m respectively. Notice that, the larger the average gap, the greater the gap size dispersion. Indeed, for $\beta = 1$ the standard deviation is equal to 0 traces (isolated missing traces

Table 5.1: *Average* SNR *[dB] achieved on gathers belonging to $\mathcal{D}_E$, for different values of α and β.*

α \ β	1	2	3
10	38.4	25.3	21.9
30	32.8	21.7	18.1
50	29.9	18.7	15.7

only); on the contrary, for $\beta = 2$ and $\beta = 3$ the standard deviations are $\sigma = 1.14$ and $\sigma = 2.44$ traces, respectively. For instance, in the datasets under examination, $\beta = 3$ provides a maximum gap up to 30 traces (corresponding to 375m), which simulates a quite large physical obstacle.

Table 5.1 depicts the average results achieved by the *U-net* on the evaluation set. Notice that, the larger the burst length, the lower the resulting SNR. This enlightens the need of further investigations for interpolating bursts of many traces: as a matter of fact, as the size of the group of adjacent missing traces increases, the ability of the network in reconstructing the unknown samples diminishes. Nonetheless, notice that even in the worst case, i.e., $(\alpha, \beta) = (50, 3)$, the *U-net* is able to maintain acceptable reconstruction performances.

Interpolation by transfer learning

In order to test the robustness of the proposed method in interpolating missing data, we generate two further synthetic datasets, exploiting the very same acquisition geometry and model of the dataset previously presented, but with different sampling rates 4ms and 8ms. The goal of this experiment is to check if the *U-net* architecture, when trained on data sampled every 6ms, is able to reconstruct differently-sampled data. This is an example of the well-known transfer learning strategy [150]. Namely, it corresponds to analyzing the performances of one network which has already been trained over a dataset having different features from the testing one.

To this purpose, we propose to select as test case the uniform missing traces framework, randomly deleting the 30% of traces from these new datasets. Then, we evaluate the reconstruction results on gathers belonging to the evaluation set $\mathcal{D}_E$ of these datasets, with the difference that we exploit the network trained on the dataset sampled every $\delta_t = 6$ms.

Average results of the interpolation are shown in Table 5.2. Notice that

Table 5.2: *Average* SNR *[dB] achieved on* $\mathcal{D}_E$*, for sampling time* $\delta_t = 4$*ms and* $\delta_t = 8$*ms.*

δ_t	4ms	8ms
Train on 6ms	10.1	10
Train on δ_t	25.3	23.8

we report also the interpolation results we can achieve if following the standard training procedure, that is, training the network using data with the same sampling time of the evaluation set. Even if the difference between the results is noticeable, Figure 5.8 shows an example of $\delta_t = 8$ms data reconstruction exploiting the *U-net* trained on $\delta_t = 6$ms.

Interpolation of regularly missing traces

The last case we investigate for the synthetic dataset is that of recovering regularly missing traces, which can be considered as upsampling the acquisition geometry and has several practical implications, e.g., interpolation in the cross-line direction. This case has its own peculiarities, and it is often the most challenging interpolation problem, especially when the dips are aliased. As a matter of fact, methods based on sparse transformations and low-rank constraints have limited application in this case because of the strong spatial aliased energies. We want to test the conjecture that network training extracts some high-level characteristics of seismic data which are more robust to alias than linear events or low rank assumptions, therefore implicitly including an anti-aliasing strategy.

We consider two scenarios. The first one consists in training the *U-net* with new data, generated by regularly deleting the 50% of the shot gathers traces, and then following the same procedure previously shown for the shot gather reconstruction. The second solution emulates the transfer learning technique presented above: we use the network trained on data with 50% randomly missing traces to reconstruct the shot gathers with 50% regularly missing traces.

The achieved SNR are 30dB and 22.8dB for the first and second solution, respectively. Results achieved over a shot gather region characterized by steep dips are shown in Figure 5.9. Notice that, by training and testing the *U-net* with the same data, the reconstruction error is very low, while with the transfer learning technique the error contains again some signal content. Nonetheless, this is an expected results and both errors seem acceptable at visual inspection. Indeed, we select as a reference a recent in-

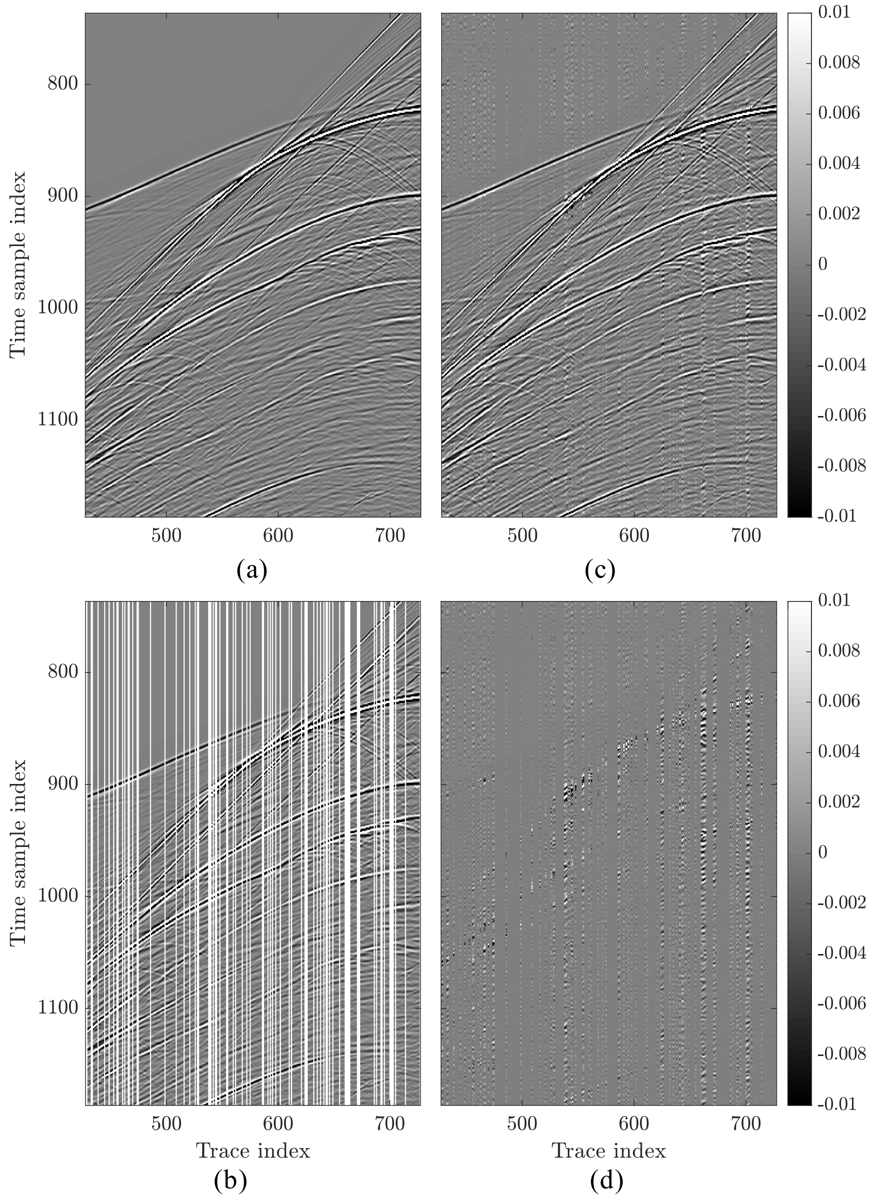


Figure 5.8: *Example of transfer learning data interpolation. (a) depicts the original gather* $\mathbf{I}$*, sampled every* $8ms$ *and cropped in its central portion with size* 450×300*; (b) shows the corrupted gather, with* 30% *of randomly distributed missing traces; (c) reports the reconstructed gather* $\hat{\mathbf{I}}$*, exploiting the* U-net *trained on data sampled every* $6ms$*; (d) shows the reconstruction error, i.e., (c) - (a).*

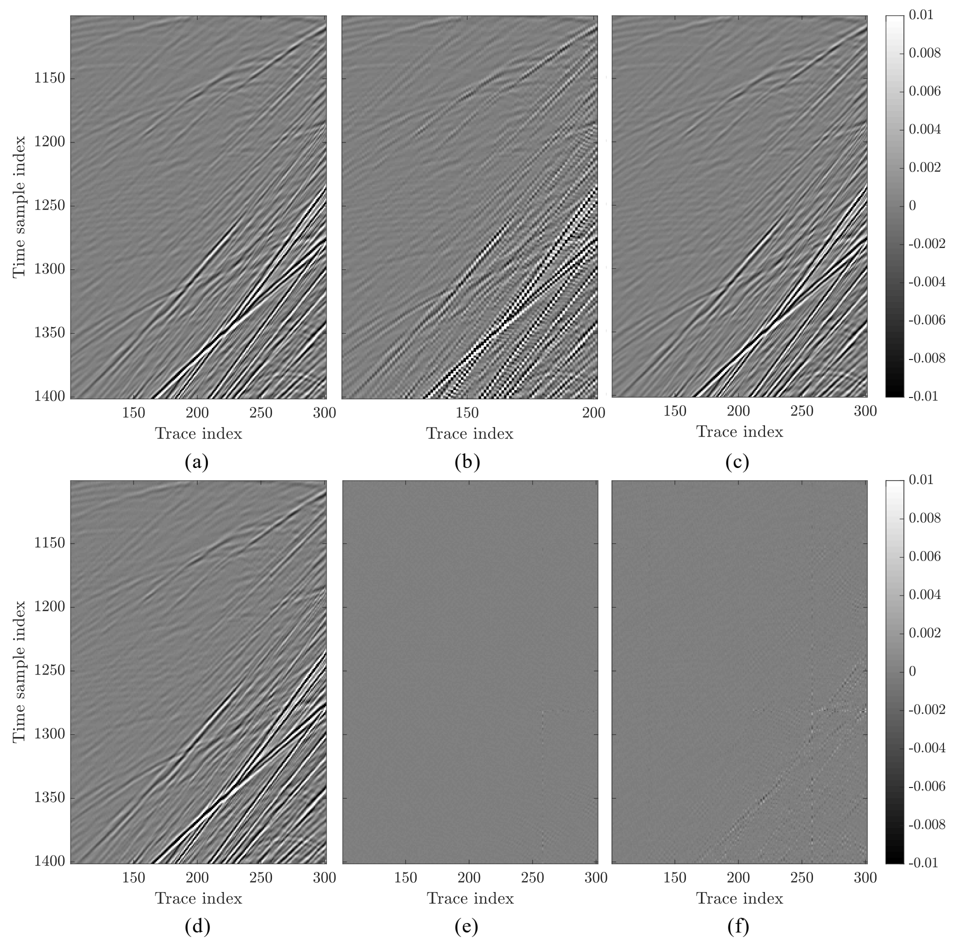


Figure 5.9: *Interpolation of regularly missing traces. (a) depicts the original gather, cropped around samples (1100 −1400, 100 −300); (b) shows the corrupted gather which contains half the original traces; (c) depicts the reconstructed gather by training the* U-net *on data with regularly missing traces; (d) reports the reconstructed gather by transfer learning; (e) shows the reconstruction error of the standard strategy, i.e., (c) - (a); (f) is the reconstruction error of the transfer learning approach, i.e., (d) - (a).*

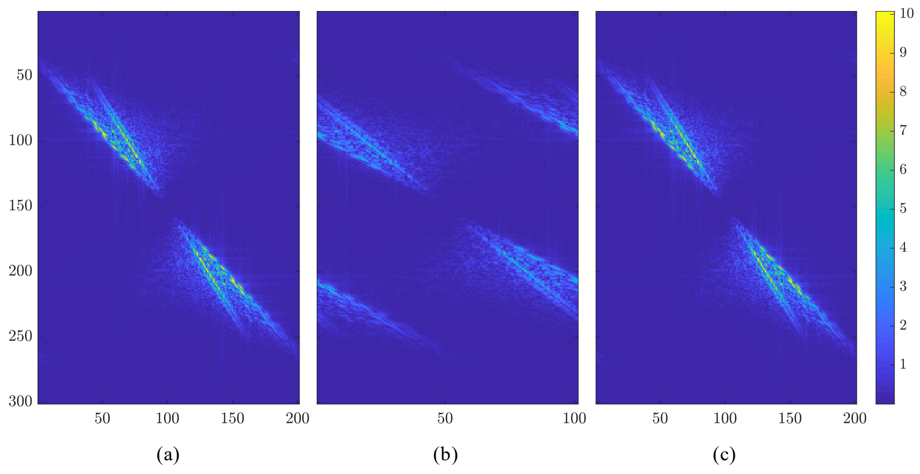


Figure 5.10: *Absolute value of Fourier spectrum of shot gathers for the problem of interpolation of regularly missing traces. (a) depicts the spectrum of the original gather cropped around samples (1100 – 1400, 100 – 300); (b) shows the spectrum of the corrupted gather which contains half the original traces; (c) depicts the spectrum of the reconstructed gather by training the* U-net *on data with regularly missing traces.*

dustrial software based on f-x deconvolution which does not require training data (hence can be considered more or less analogous to the transfer learning technique), and it achieves S/N $= 22.7$dB on the same data.

For the sake of clarity, Figure 5.10 includes the absolute value of the Fourier spectrum computed for the original shot gather, for the corrupted one, and for the estimated one by means of the first proposed strategy. It is worth noting that the alias introduced in the corrupted shot gather can be deleted by our reconstruction method.

Field data

In this section, we apply the *U-net* for reconstructing corrupted real seismic data and compare the result with those obtained by us in one preliminary work regarding interpolation only [21]. To this purpose, we exploit as field data the well known Mobil Avo Viking Graeben Line 12 dataset [159]. Specifically, this dataset consists of 1001 marine shot gathers. Each gather is composed of 128 traces of 1408 time samples, with temporal sampling of 4ms and receiver sampling of 25m.

In order to compare our results with those obtained in [21], we simulate a seismic acquisition with a uniform distribution of randomly missing traces. Therefore, for each acquired gather $\mathbf{I}$, we randomly delete the $H\%$ of its traces, $H \in \{10, 30, 50\}$, obtaining a scattered sampled gather $\bar{\mathbf{I}}$.

Table 5.3: *Average* SNR *[dB] achieved on gathers from dataset in [159].*

H	10	30	50
New *U-net*	25.7	20.5	16.7
Old *U-net*	22.5	15.1	10.2

Following the same rationale of the synthetic example, we split each dataset into 250 gathers for training and validation and leave the remaining to evaluation set. Then, in order to achieve a similar number of patches per gather (i.e., 135 in the synthetic case), we extract 129 patches with size 128×128, overlapped only on the temporal dimension, specifically with patch-stride of 10 samples. Notice that, in this case, the presence of patch overlap does not cause issues in memory usage, as the number of samples entering the network is similar to the chosen configuration for the synthetic example.

Results obtained on the evaluation dataset $\mathcal{D}_E$ are reported in Table 5.3, while Figure 5.11 shows an example of gather reconstruction where 50% of traces is missing. It is noticeable the improvement in performances of the proposed architecture, This achievement is due to the specific changes performed on the *U-net* architecture as described in the *U-net* implementation Section. As a matter of fact, even with a reduced amount of gathers for training and validation (i.e., 25% of the whole dataset instead of 75%), the resulting SNR always exceeds the past performances.

In order to build a preliminary evaluation of the results which would be obtained by seismic imaging algorithms on data reconstructed by *U-net*, we migrate the sections of the original Viking Graben dataset, of the corrupted dataset with 50% randomly missing traces and of the dataset reconstructed by the proposed methodology. As a comparison, we migrate also the dataset reconstructed using the nonlinear shaping regularization method proposed by [132]. The images depicted in Figure 5.12 show that, in case of data reconstructed with *U-net*, noise is attenuated and loss of continuity is almost perfectly recovered in the migrated section. This occurs also for the reconstruction of [132], however, contrarily to that of *U-net*, the migrated section shows some spurious events which are not present in the original image.

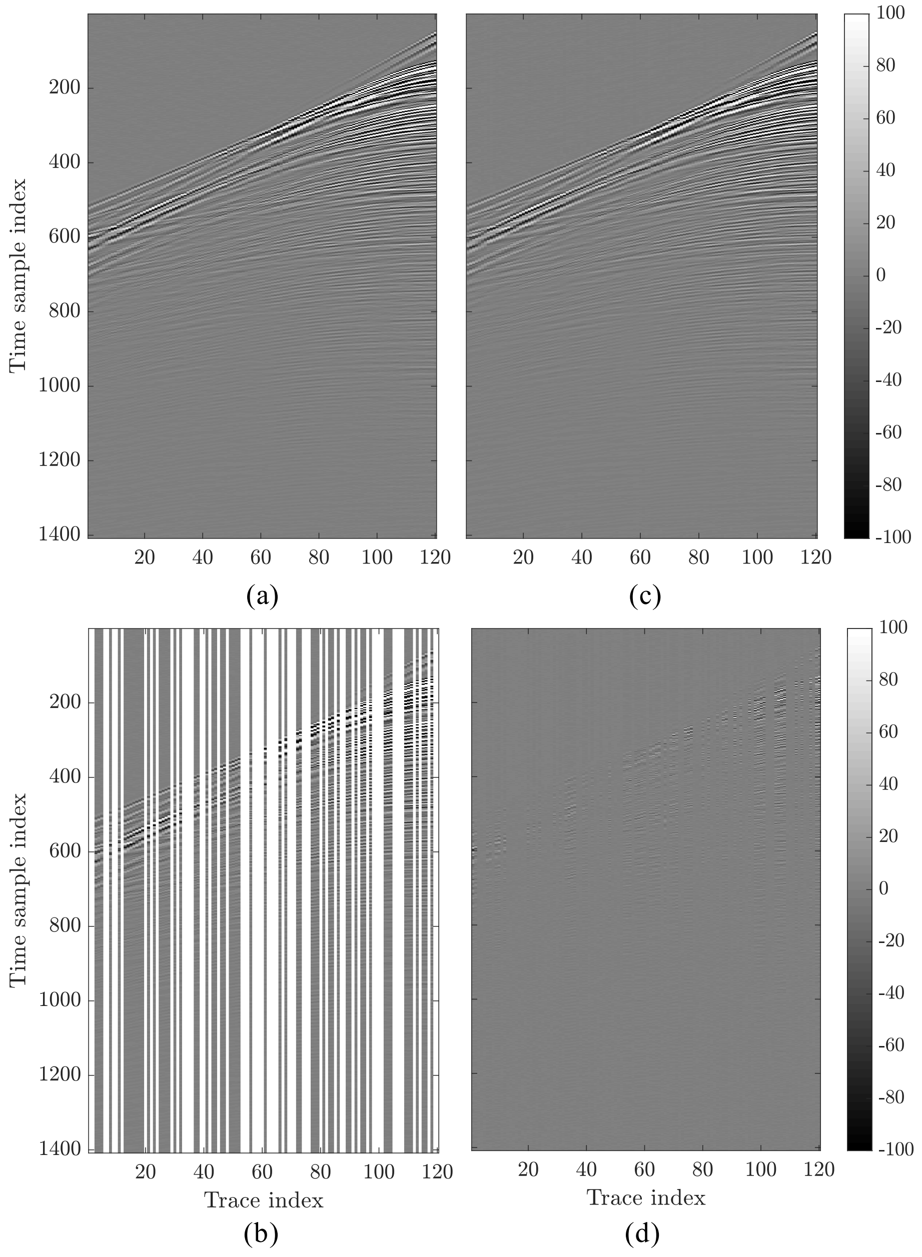


Figure 5.11: *Example of data interpolation, considering one gather of the field dataset [159]. (a) depicts the original gather* $\mathbf{I}$*; (b) reports the corrupted gather* $\bar{\mathbf{I}}$*, with* 50% *of randomly missing traces; (c) shows the reconstructed gather* $\hat{\mathbf{I}}$*; (d) reports the reconstruction error, i.e., (c) - (a).*

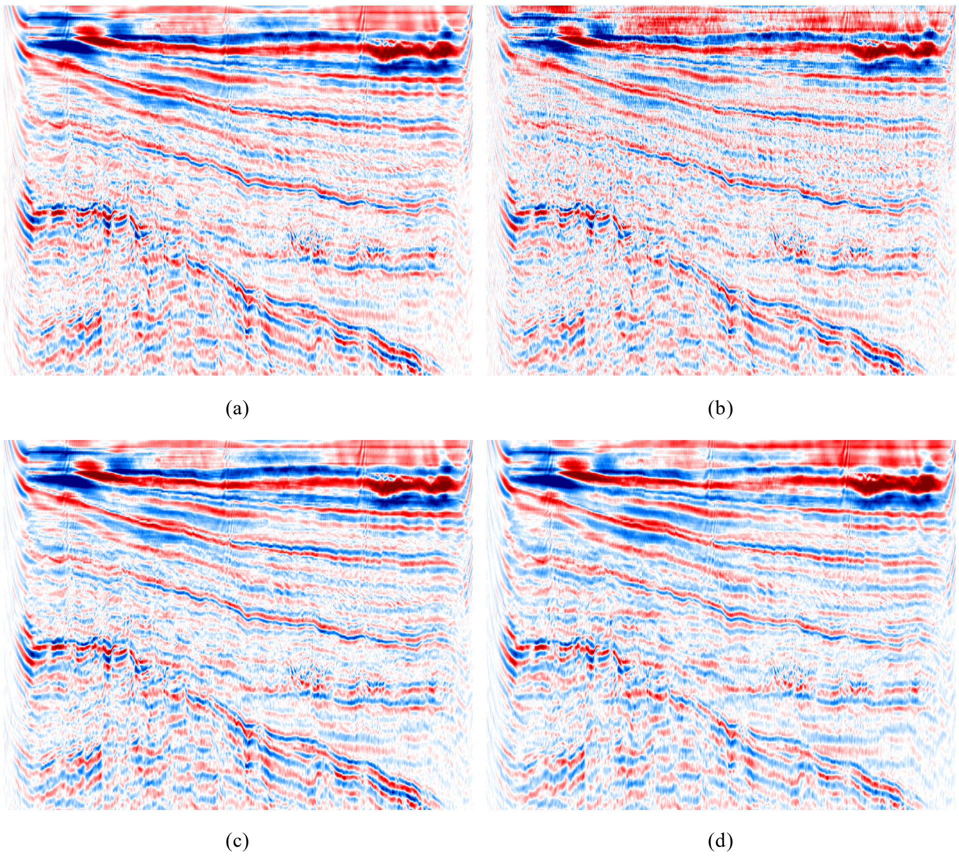


Figure 5.12: *Kirchhoff migrated section of the Viking Graben dataset: (a); migration of original data; (b) migration of corrupted data; (c) migration of data reconstructed by* U-net*; (d) migration of data reconstructed by the method proposed in [132].*

5.3.3 Problem 2: Denoising of corrupted gathers

In this section we report the results of the numerical experiments related to denoising of corrupted gathers. We consider two kinds of additive noise, the standard additive white Gaussian noise (AWGN), and a spike-like noise. Concerning the patch extraction methodology to be applied, we select exactly the same strategy of that presented in Section related to interpolation of missing traces. As a matter of fact, the presence of randomly missing traces as well as the additive random noise can be seen as two generic kinds of gather corruption, which can be tackled by the *U-net* in a similar way.

AWGN noise model

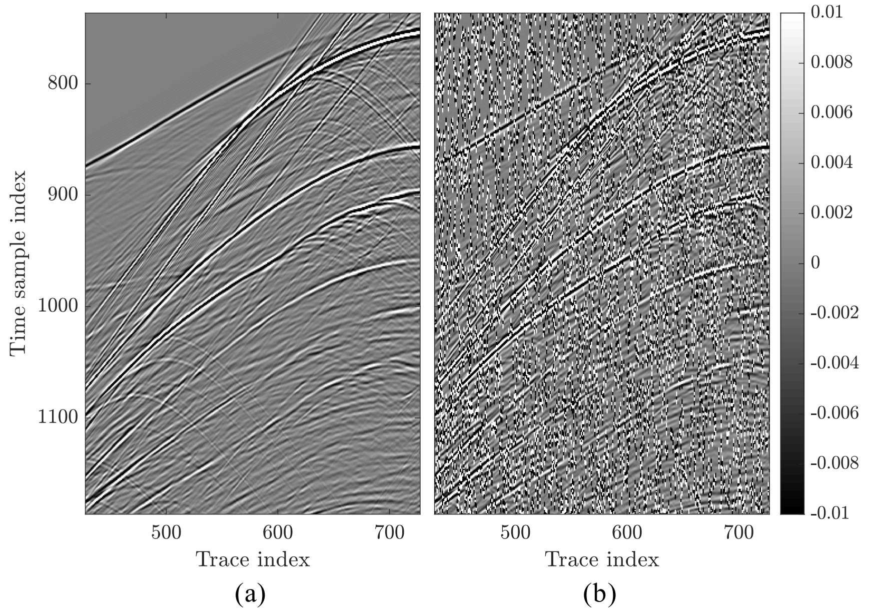
In order to test our method over a plurality of signal to noise ratios (SNR), we add white gaussian noise for achieving $\mathrm{SNR} = S \in \{-3, 0, 3\}$dB, defined as the ratio between the signal and noise power. The average results obtained on shot gathers belonging to the evaluation set $\mathcal{D}_E$ are the following: $\mathrm{SNR} = 12.8$dB, 14.4dB and 16.3dB, correspondent to increasing values of S. Indeed, these results can be considered an upper bound for the achievable performances of the proposed strategy in realistic scenarios. As a matter of fact, they are obtained on the assumption to have clean data available for the training phase, which is never the case for field acquired data.

Spike-like noise model

Pre-stack seismic data can be affected by different types of random noise coming from various sources, such as wind motion, poorly planted geophones or electrical noise, most of these being far more complex than simple AWGN. For instance, some of these seismic noises exhibit spike-like characteristics [160] and are lately gaining growing interest, as they strongly affect the processing of simultaneous source data acquired from recent seismic surveys [161].

Therefore, we propose to use our network for denoising data corrupted by additive spike-like noise. In order to simulate this noise, we add spiky noise with variable density $d\%$, namely the percentage of corrupted samples in one gather. In particular, the binary values of this noise are set to the minimum and maximum values of the original uncorrupted data. Then, we convolve each noise trace with a Ricker wavelet having the same central frequency of the data (i.e., 27Hz) and unit energy. This way, we generate two corrupted datasets, corresponding to $d \in \{1, 3\}$.

Figure 5.13 shows an example of spike-like corruption denoising for $d = 3$. It is noticeable that, even if the corrupted image visually undergoes a strong degradation, the reconstructed one presents almost all the features of the original data. Moreover, Figure 5.13(d) reports the residual error between corrupted and reconstructed gather. It is worth noting that a large portion of the useful signal remains unaltered, except for very reduced areas. This trend is confirmed by Table 5.4, which reports the average results achieved on set $\mathcal{D}_E$.



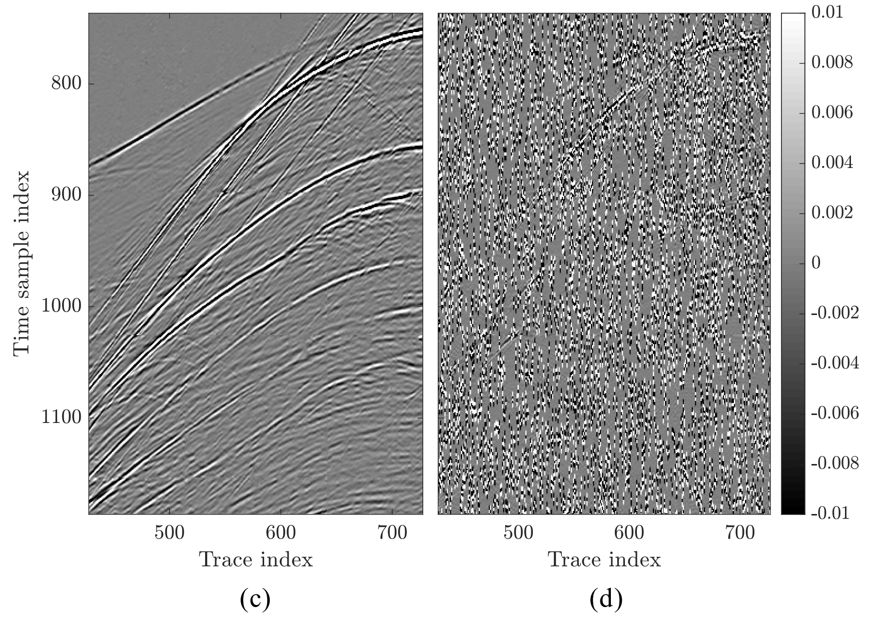


Figure 5.13: *Example of spike-like noise attenuation. (a) reports the original gather; (b) reports the corrupted gather, with noise density $d = 3$; (c) shows the reconstructed gather; (c) shows the residual error, i.e., (b) - (c).*

Table 5.4: *Initial* SNR *[dB] on corrupted gathers and final* SNR *[dB] achieved by spike-like denoising.*

d	1	3
Initial	-34.8	-39.6
Final	16.8	12.3

Towards standard denoiser emulation

In order to highlight the *U-net*-based method versatility, we propose to exploit our denoising strategy as an emulator of some well known standard denoising algorithms. In particular, we select two noise attenuation strategies, namely a Wavelet based denoising [162] and a f-x deconvolution method included in a recent production software. To test the denoising performances, we use the datasets corrupted by AWGN with $\text{SNR} = S \in \{-3, 0, 3\}$dB.

Initially, we process the whole datasets through the aforementioned standard denoising algorithms. Through this operation, we are generating denoised data which can be more or less considered in the same way as clean uncorrupted gathers.

In a second phase, we train our network in a slightly different way than the approach shown in *U-net* training Section. Indeed, we train the *U-net* substituting to ground truth gathers those obtained through denoising by Wavelet or by the industrial f-x deconvolution. Thus, the training step includes pairs of noisy gathers and gathers denoised by standard algorithms. Eventually, we evaluate denoising results on shot gathers belonging to $\mathcal{D}_E$, comparing reconstructed gathers with the original ones, as described in (5.5). Specifically, we compute the average results for *U-net* and for standard denoising algorithms as well.

From results depicted in Table 5.5 it is quite evident that performances of *U-net* are comparable with those achieved through the denoising algorithm used for training, showing that *U-net* is able to mimic their performances. Moreover, the proposed method has a further advantage, which is the low computational effort in denoising a generic gather. As a matter of fact, if a limited amount of time is needed for training the network model parameters, the evaluation phase is very efficient: the optimized production solution and the Wavelet denoising take respectively the same time and 25% more than the time required by our strategy (i.e., approximately 0.3 seconds) for estimating each denoised gather.

These results pave the way towards one potential application of our

Table 5.5: *Average* SNR *[dB] on gathers belonging to $\mathcal{D}_E$, training the* U-net *on gathers denoised by Wavelet (a) and by the industrial software based on f-x deconvolution (b).*

S	0	3	
U-net	4.8	5.7	6.9
Wavelet	4.7	5.7	6.9

(a)

S	-3	0	3
U-net	7	8.6	10.3
f-x deconv	7	8.7	10.3

(b)

method in realistic situations. Indeed, as previously stated, having clean gathers available for the training phase is not the case when dealing with real data. Furthermore, denoising field acquisitions often require complex and computationally expensive algorithms.

In order to overcome these issues, we recommend our strategy as a viable alternative to many standard denoising algorithms. Specifically, when a large field dataset is available, the following algorithm can be applied:

1. randomly select a subset of the acquired shot gathers;
2. perform an accurate and computationally expensive denoising on the selected shot gathers;
3. train the *U-net* on the selected pairs of acquired/denoised gathers;
4. make use of the trained *U-net* to denoise the remaining data.

After a certain dimension of the dataset, due to the fixed computational cost for denoising the selected subset and training the *U-net*, the application of the *U-net*-based denoising becomes computationally cheaper than denoising the whole dataset with the standard noise attenuation algorithm. Indeed, the computational advantage of *U-net* increases with the dimension of the dataset and the complexity of the denoising algorithm.

5.3.4 Complete problem: Joint interpolation and denoising

The last situation we propose is the more realistic case of study, implying additive noise corruption jointly with missing traces. We investigate two cases of study: the former exploits the same synthetic dataset of all the previous experiments, while the latter uses a different dataset, comparing our method with state-of-the-art techniques.

Table 5.6: *Average* SNR *[dB] achieved on gathers belonging to $\mathcal{D}_E$, for each dataset extracted from BP-2004 [157].*

$H \setminus S$	-3	0	3
10	12.2	13.8	15.6
30	11.5	12.9	14.4
50	10.4	11.6	12.9

AWGN and uniform missing traces

In order to investigate if the proposed method is able to retrieve the original synthetic data, we consider the presence of AWGN and uniformly distributed missing traces. Likewise previously done, we add noise leading into $S \in \{3, 0, -3\}$dB and delete a percentage $H \in \{10, 30, 50\}$ of the available data traces for simulating seismic acquisition with irregular receiver sampling and missing traces. This way, we generate 9 different datasets, corresponding to various combinations of additive noise and missing traces. Table 5.6 resumes the average results obtained on shot gathers belonging to the evaluation set $\mathcal{D}_E$, considering all possible combinations of missing traces and additive noise variances.

Comparison with a recent data-driven method

To compare our strategy with a recent learning-based algorithm, we consider the Double-Sparsity Dictionary Learning method proposed by [144] and one strategy based on fixed dictionary transform used as baseline in [144], i.e., the Curvelet method. In order to perform a fair comparison, we reproduce exactly the same synthetic example provided in [144].

Specifically, the dataset is extracted from the BP-1997 benchmark [163] and includes 385 shot gathers, considering the last 240 receivers (taking the source as reference) with 384 samples/trace. We add noise and delete some traces in the dataset following the procedure described therein. In the first stage, as done in [144], we normalize the range of each trace to 1. Then, white gaussian noise is generated, low pass-filtered with a cut-off frequency of 30Hz and finally added to the traces. We perform the same numerical experiments proposed in [144], testing plenty of noise standard deviations $\sigma \in \{0.05, 0.10, 0.15, 0.20, 0.25\}$ and missing traces' percentages $H \in \{10, 20, 30, 33, 40, 50, 60\}$.

For what concerns the training phase, we randomly select 250 shot gathers for training and validation (split in $75\% - 25\%$). In test phase, we process exactly the same image of that used in [144], namely the first shot gather in the dataset. Note that, in order to be fair, we never use this shot gather in the training phase.

During training, we extract from each image 153 overlapping patches with size 128×128 and stride $(16, 14)$ along rows and columns, respectively. This operation has been done in order to achieve training and validation sets with similar size (concerning the number of patches) to the previously shown situations. Notice that we consider exactly the same training-validation-evaluation procedure of that depicted in *U-net* training and deployment Sections.

For comparing the results, we use the evaluation metrics proposed in [144], namely the peak signal to noise ratio (PSNR), defined as

$$\mathrm{PSNR} = 10\ \log_{10} \frac{s_{\max}}{\sigma^2(\mathbf{I} - \hat{\mathbf{I}})}, \tag{5.7}$$

being $s_{\max}$ the dynamic range of the clean signal, thus fixed to 1.

Figure 5.14(a) shows the original shot gather without noise added and missing traces (i.e., the ground truth of the experiment). The corrupted version of the gather with 33% of missing traces and $\sigma = 0.1$ is shown in Figure 5.14(b).

Figures 5.14(c)-(d) show the recovered gathers obtained with double-sparsity dictionary learning and *U-net*, respectively. We can notice that there are some events which are well reconstructed by the *U-net* while are missing in the retrieved shot gather via double-sparsity dictionary learning. Specifically, Figures 5.14(e)-(f) show the error panels (i.e., the difference between the recovered images (c) and (d) and the ground truth (a)) for the results obtained with the state-of-the-art technique and *U-net* respectively. It is quite evident that the error corresponding to double-sparsity dictionary learning is more affected by residual coherent events, meaning that those are not correctly recovered. These qualitative considerations are confirmed by the corresponding PSNR values: 32.1 dB for double-sparsity dictionary learning and 33.7 dB for *U-net*. Clearly, for both methods the reconstruction error is not optimum and there is still room for improvement. Nonetheless, it is worth noting that the limited size of the shot gathers (only 240×384, versus the 1152×1920 of BP-2004 benchmark and the 128×1408 of Mobil Avo Viking Graeben Line 12 dataset) could potentially undermine the ability of *U-net* to learn how to describe the complex features of the clean data without modeling noise and missing data. Indeed,

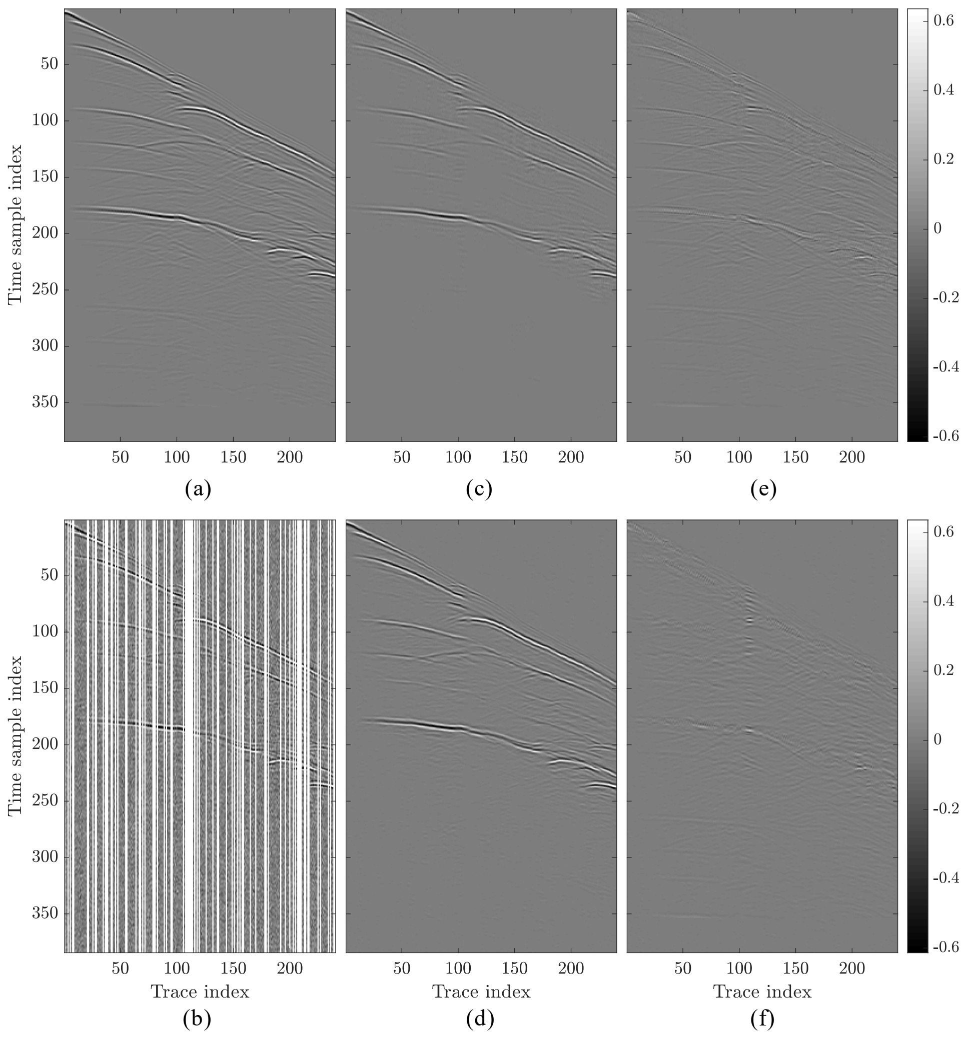


Figure 5.14: *Shot gather with* 33% *of missing traces and* $\sigma = 0.1$. *(a) shows the original gather; (b) depicts the corrupted version; (c) shows the recovered gather obtained with double-sparsity dictionary learning,* $\mathrm{SNR} = 32.1$ *dB; (d) shows the recovered gather obtained with* U-net, $\mathrm{SNR} = 33.7$ *dB; (e) illustrates the reconstruction error of double-sparsity dictionary learning, i.e., (c) - (a); (f) depicts the reconstruction error of our method, i.e., (d) - (a).*

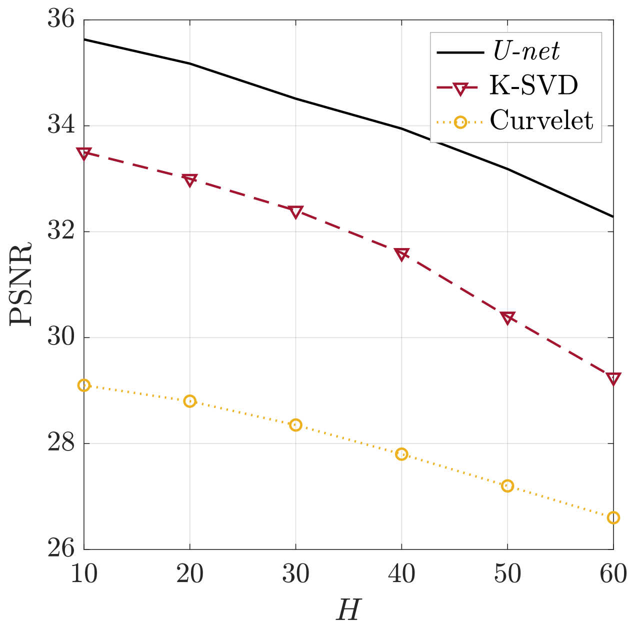


Figure 5.15: *Results of different reconstruction methods by varying the missing traces ratio H and for $\sigma = 0.1$.*

to be trained, CNNs usually need a substantial amount of data for achieving acceptable performances. Whether more data samples were accessible, we expect the performances to improve accordingly.

Figure 5.15 displays the performances of different reconstruction methods by varying the missing traces ratio and selecting $\sigma = 0.1$. In particular, we compare results reported in [144] with our results, averaged over 100 different realizations of the column pattern used for randomly deleting the traces. It is noticeable that we significantly outperform both the dictionary learning-based method and the Curvelet-based, gaining an average of 2.4 dB with respect to the former strategy and 6.1 dB to the latter one.

Figure 5.16 reports the achieved results for a plurality of noise standard deviations. The performances of the *U-net* are significantly superior than those of dictionary learning-based strategy, in all the examined cases.

Moreover, our method reveals to be more robust in presence of strong noise. As a matter of fact, as noise standard deviation σ increases, the curves related to state-of-the-art method decay in a worse fashion than ours, to the point that we can achieve $\text{PSNR} = 30.3$ dB for $H = 50$ and $\sigma = 0.25$, against the 26.7 dB of the dictionary learning-based technique.

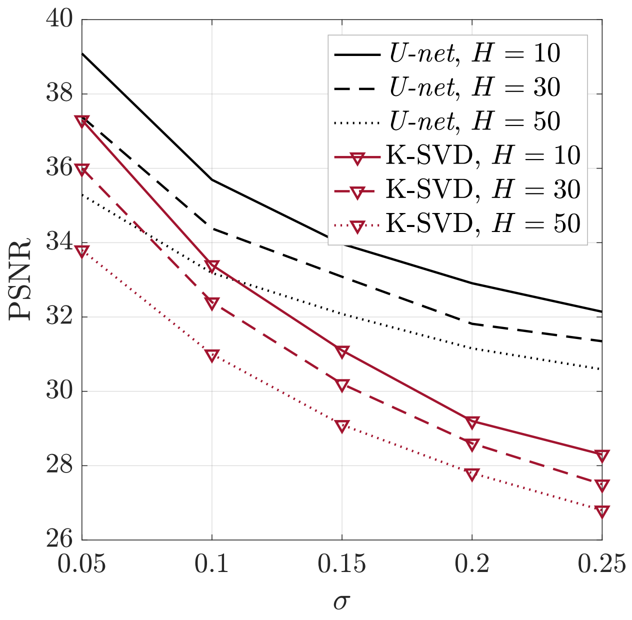


Figure 5.16: *Results of* U-net *and double-sparsity dictionary learning reconstruction methods by varying the missing traces ratio* $H = \{10, 30, 50\}$ *and for* $\sigma = 0.05 : 0.05 : 0.25$.

5.4 Conclusions

In this chapter, we propose a method for reconstruction of corrupted seismic data, focusing on noise attenuation and interpolation of missing pre-stack data traces in the shot-gather domain. In particular, we consider random noise cases with different statistics and a variety of missing traces distributions. Our approach makes use of a convolutional neural network architecture for interpolation and denoising of 2D shot gathers, showing performance improvements, in terms of SNR, with respect to recent solutions for joint denoising and interpolation.

Results achieved on controlled synthetic experiments demonstrate that the proposed method is a promising strategy for seismic data pre-processing. The method is capable of effectively and efficiently restore 2D corrupted data, and it is also able to deal with the task of spatially upsampling the shot gathers. Moreover, once the network training procedure is completed, processing data with our strategy is also quite efficient in terms of computational effort.

We examine also the potential application of our methodology on field data for production environments. In this situation, it is interesting to notice that the proposed algorithm can be used also to emulate the effect of more time consuming classical data pre-processing strategies.

CHAPTER 6

Conclusions

This thesis investigates the source device identification problem tackling the issue from forensics and counter-forensics perspectives.

Chapters 2 and 3 present forensics investigations on source device identification for images and video sequences, while Chapter 4 focuses on counter-forensics for image device anonymization.

More in detail, Chapter 2 develops a novel methodology to deal with source device identification on images. We propose a CNN-based strategy with the specific purpose of keeping computational complexity at bay in case a large amount of provenance devices have to be tested over a query image. We investigate different network architectures, training strategies and loss functions, motivated by two primary goals: (i) saving important storage and time; (ii) improving the accuracy achieved by statistical approaches. The proposed method has been tested over more than 80 different devices taken from Dresden and Vision datasets. Our experimental campaign shows that less query image content and reduced computation time are needed to provide comparable accuracies with state-of-the-art, electing our strategy as a viable alternative to standard model-based approaches.

In this vein, a forthcoming application could be the investigation of source device identification among social networks images. However, the multi-processing chain images usually undergo when circulating over Internet could strongly hinder the performances of our CNN-based method, as visual content is often severely compressed and tampered with. In the last part of Chapter 2, we analyze the compression problem and propose a method to estimate the number of JPEG compressions of an image. Exploiting the Task-driven Non-negative Matrix Factorization model, we consider up to four compressions with multiple quality factors. We test the method on different datasets showing results that can outperform state-of-the-art.

In Chapter 3 we tackle source device identification on video sequences. We thoroughly analyze the video stabilization technology, showing how this can impede the video-camera attribution problem unless suitable countermeasures are considered. Modeling the stabilization mechanism as a similarity transformation applied on video frames, we propose two solutions to estimate the reference fingerprint of a stabilized device and three alternative strategies to identify the source device of a query stabilized video. To perform these tasks, we investigate global optimization algorithms and Fourier Mellin transform. Furthermore, we provide an interesting insight about the stabilization effect on the first acquired video frame. Up to now, we noticed this frame is usually not stabilized thus can be exploited to increase identification performances, however future scenarios may include video sequences whose totality of frames is motion compensated.

An application of source device identification on videos is described in the last part of Chapter 3. Specifically, given a video compilation composed by a plurality of video sequences coming from unknown devices, we aim at blind detecting and localizing the splicing points. To this purpose, we leverage sensor-related noise traces extracted from frames to group together the portions of compilation generated from the same camera.

Chapter 4 is dedicated to counter-forensics investigations on images. Precisely, we tackle the image anonymization problem, that is, removing from images the traces related to the original source camera. To this purpose, we modify the image pixel content fixing two objectives: (i) lowering the cross-correlation test with the camera PRNU; (ii) maintaining a high image quality level.

In the initial part of this chapter, we propose a model-based approach which deletes some image pixels and inpaints them by means of regularization and denoising techniques. Then, a solution leveraging CNNs is presented as well. Opposite to the inpainting-based strategy which is *blind*, namely it does not require the PRNU to remove being known at analyst side, the CNN strategy trains a neural network using this specific PRNU. It is worthy of note the unfamiliar use we make of CNNs, seeing the network as a parametric operator to overfit on each pair of input image-PRNU. Even though this anonymization approach can be unmasked by an informed analyst thus still requires efforts to achieve actual effectiveness, the proposed method allows to keep computational complexity at bay and poses new challenges towards deployment of completely data-driven and automatic anonymization systems.

Chapter 5 includes further applications of interpolation and denoising strategies on seismic images. Precisely, from an image processing perspective, 2D pre-stack seismic traces in the shot gather domain can be viewed as double floating point format images without loss of generality. Given these premises, we exploit a particular CNN architecture known as *U-net* for extracting a compact representation of these images, allowing to reconstruct corrupted data with missing traces and additive random noise. Our strategy proves to be effective also in dealing with transfer learning procedures and with the issue of spatially upsampling the shot gathers.

Overall, the research done in this thesis paves the way for a set of future perspectives. Being able to solve the image source device identification problem with CNNs opens new challenges on video-camera attribution as well. Since video stabilization introduces geometrical transformations leading to frame misalignment, CNN-based strategies could be developed as alternative efficient and faster approaches to those presented in Chapter 3. For instance, CNNs could be trained using a similar paradigm of that shown in Chapter 2 to infer the coherence between a query stabilized sequence and a candidate device. Given a set of frames recorded by the same camera, CNNs could be exploited to compute a reliable reference device fingerprint. In other words, CNNs could act as a video de-stabilizer: given a stabilized video, the network should learn how to follow the subtle sensor traces left on frames to correctly realign them, in such a way as to delete the effect of stabilization.

Given the unbridled growth of visual data flowing over the Internet, automatic and data-driven approaches are rapidly becoming the most sought-after methods regarding forensics tasks. Developing model-based solutions tailored to the specific problem might be yet ineffective when dealing with large and heterogeneous databases. In this vein, it is clear that one valid road for dealing with huge amount of resources is represented by deep learning strategies. CNNs are purposely designed with the goal of learning the best data representation with high effectiveness and relatively low computational effort, thus arousing forensics research in this direction.

Bibliography

[1] S. Bradshaw and P. N. Howard, “The global disinformation disorder: 2019 global inventory of organised social media manipulation,” *Oxford, UK: Project on Computational Propaganda*, 2019.

[2] S. C. Woolley and P. N. Howard, *Computational propaganda: political parties, politicians, and political manipulation on social media*. Oxford University Press, 2018.

[3] S. Bradshaw and P. N. Howard, “Challenging truth and trust: A global inventory of organized social media manipulation,” *The Computational Propaganda Project*, 2018.

[4] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, “Vision of the unseen: Current trends and challenges in digital image and video forensics,” *ACM Computing Surveys (CSUR)*, vol. 43, pp. 26:1–26:42, 2011.

[5] C. Wardle and H. Derakhshan, “Information disorder: Toward an interdisciplinary framework for research and policy making,” *Council of Europe Report*, vol. 27, 2017.

[6] H. Farid, “Exposing digital forgeries in scientific images,” in *Proceedings of the 8th workshop on Multimedia and security*. ACM, 2006, pp. 29–36.

[7] I. Korshunova, W. Shi, J. Dambre, and L. Theis, “Fast face-swap using convolutional neural networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3677–3685.

[8] D. Güera and E. J. Delp, “Deepfake video detection using recurrent neural networks,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[10] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008.

[11] L. Bondi, P. Bestagini, F. Pérez-González, and S. Tubaro, “Improving PRNU compression through preprocessing, quantization, and coding,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 608–620, 2019.

[12] M. Iuliani, M. Fontani, D. Shullani, and A. Piva, “Hybrid reference-based video source identification,” *Sensors*, vol. 19, no. 3, 2019.

[13] S. Taspinar, M. Mohanty, and N. Memon, “Source camera attribution using stabilized video,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016, pp. 1–6.

[14] S. Mandelli, D. Cozzolino, P. Bestagini, L. Verdoliva, and S. Tubaro, “Cnn-based fast source device identification,” *arXiv preprint arXiv:2001.11847*, 2020.

[15] S. Mandelli, N. Bonettini, P. Bestagini, V. Lipari, and S. Tubaro, “Multiple jpeg compression detection through task-driven non-negative matrix factorization,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2106–2110.

[16] S. Mandelli, P. Bestagini, L. Verdoliva, and S. Tubaro, “Facing device attribution problem for stabilized video sequences,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 14–27, 2020.

[17] S. Mandelli, P. Bestagini, S. Tubaro, D. Cozzolino, and L. Verdoliva, “Blind detection and localization of video temporal splicing exploiting sensor-based footprints,” in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1362–1366.

[18] S. Mandelli, L. Bondi, S. Lameri, V. Lipari, P. Bestagini, and S. Tubaro, “Inpainting-based camera anonymization,” in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1522–1526.

[19] N. Bonettini, L. Bondi, D. Güera, S. Mandelli, P. Bestagini, S. Tubaro, and E. J. Delp, “Fooling prnu-based detectors through convolutional neural networks,” in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 957–961.

[20] J. Lukas, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[21] S. Mandelli, F. Borra, V. Lipari, P. Bestagini, A. Sarti, and S. Tubaro, “Seismic data interpolation through convolutional autoencoder,” in *88th International Annual Meeting*. SEG, Expanded Abstracts, 2018, pp. 4101–4105.

[22] S. Mandelli, V. Lipari, P. Bestagini, and S. Tubaro, “Interpolation and denoising of seismic data using convolutional neural networks,” *arXiv preprint arXiv:1901.07927*, 2019.

[23] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, “Source digital camcorder identification using sensor photo response non-uniformity,” in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505, 2007, p. 65051G.

[24] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, “First steps toward camera model identification with convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259–263, 2016.

[25] A. Tuama, F. Comby, and M. Chaumont, “Camera model identification with the use of deep convolutional neural networks,” in *2016 IEEE International workshop on information forensics and security (WIFS)*. IEEE, 2016, pp. 1–6.

[26] D. Cozzolino and L. Verdoliva, “Noiseprint: a CNN-based camera model fingerprint,” *IEEE Transactions on Information Forensics and Security, in press*, 2019.

[27] M. C. Stamm, M. Wu, and K. R. Liu, “Information forensics: An overview of the first decade,” *IEEE access*, vol. 1, pp. 167–200, 2013.

[28] M. Kirchner and T. Gloe, “Forensic camera model identification,” *Handbook of Digital Forensics of Multimedia Data and Devices*, pp. 329–374, 2015.

[29] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva, “VISION: a video and image dataset for source identification,” *EURASIP Journal on Information Security*, vol. 2017, no. 1, p. 15, 2017.

[30] M. Grundmann, V. Kwatra, and I. Essa, “Cascaded camera motion estimation, rolling shutter detection, and camera shake detection for video stabilization,” Feb. 6 2018, uS Patent 9,888,180.

[31] J. Lukas, J. Fridrich, and M. Goljan, “Determining digital image origin using sensor imperfections,” *Image and Video Communications and Processing*, vol. 5685, pp. 249–260, 2005.

[32] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, “A Bayesian-MRF approach for PRNU-based image forgery detection,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 554–567, 2014.

[33] X. Lin and C. Li, “Large-scale image clustering based on camera fingerprints,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 793–808, 2017.

[34] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “Blind PRNU-based image clustering for source identification,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2197–2211, 2017.

[35] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, “Compressed fingerprint matching and camera identification via random projections,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1472–1485, 2015.

[36] M. Goljan and J. Fridrich, “Camera identification from cropped and scaled images,” in *SPIE Electronic Imaging (EI)*, 2008.

[37] X. Lin and C.-T. Li, “Large-scale image clustering based on camera fingerprints,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 793–808, 2016.

[38] Q. Phan, G. Boato, and F. D. Natale, “Accurate and scalable image clustering based on sparse representation of camera fingerprint,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1902–1916, 2018.

[39] C. Meij and Z. Geradts, “Source camera identification using Photo Response Non-Uniformity on WhatsApp,” *Digital Investigation*, vol. 24, pp. 142–154, 2018.

[40] R.Rouhi, F. Bertini, D. Montesi, and C.-T. Li, “Social network forensics through smartphones and shared images,” in *International Workshop on Biometrics and Forensics*, 2019.

[41] L. Bondi, L. Baroffio, D. Guera, P. Bestagini, E. Delp, and S. Tubaro, “First steps toward camera model identification with convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 259–263, 2017.

[42] T. Gloe and R. Böhme, “The dresden image database for benchmarking digital image forensics,” *Journal of Digital Forensic Practice*, vol. 3, pp. 150–159, 2010.

[43] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[45] W. Gao and Z.-H. Zhou, “On the consistency of auc pairwise optimization,” in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[46] S. Gultekin, A. Saha, A. Ratnaparkhi, and J. Paisley, “Mba: Mini-batch auc optimization,” *arXiv preprint arXiv:1805.11221*, 2018.

[47] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," in *Advances in neural information processing systems*, 2001, pp. 472–478.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[49] M. C. Stamm, M. Wu, and K. J. R. Liu, "Information forensics: An overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[50] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, p. 22, 2013.

[51] T. Bianchi, A. Piva, and F. Pérez-González, "Near optimal detection of quantized signals and application to JPEG forensics," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2013.

[52] M. Carnein, P. Schöttle, and R. Böhme, "Forensics of high-quality JPEG images with color subsampling," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2015.

[53] M. Barni, Z. Chen, and B. Tondi, "Adversary-aware, data-driven detection of double JPEG compression: How to make counter-forensics harder," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016.

[54] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. J. R. Liu, "Undetectable image tampering through jpeg compression anti-forensics," in *IEEE International Conference on Image Processing (ICIP)*, 2010.

[55] C. Pasquini and G. Boato, "JPEG compression anti-forensics based on first significant digit distribution," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[56] M. Kirchner and S. Chakraborty, "A second look at first significant digit histogram restoration," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2015.

[57] T. Bianchi and A. Piva, "Detection of nonaligned double JPEG compression based on integer periodicity maps," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 842–848, 2012.

[58] T. Thai, R. Cogranne, F. Retraint, and T. Doan, "JPEG Quantization Step Estimation and Its Applications to Digital Image Forensics," *IEEE Transactions on Information Forensics and Security*, vol. PP, no. 99, pp. 123–133, 2016.

[59] M. Barni, L. Bondi, N. Bonettini, P. Bestagini, A. Costanzo, M. Maggini, B. Tondi, and S. Tubaro, "Aligned and non-aligned double jpeg detection using convolutional neural networks," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 153 – 163, 2017.

[60] T. Pevny and J. Fridrich, "Detection of double-compression in JPEG images for applications in steganography," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 3, pp. 247–258, 2008.

[61] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of jpeg artifacts," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, pp. 1003–1017, 2012.

[62] Q. Wang and R. Zhang, "Double JPEG compression forensics based on a convolutional neural network," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 23, December 2016.

[63] B. Li, Y. Shi, and J. Huang, "Detecting doubly compressed JPEG images by using mode based first digit features," in *IEEE Workshop on Multimedia Signal Processing (MMSP)*, 2008.

[64] C. Pasquini, G. Boato, and F. Pérez-González, “Multiple JPEG compression detection by means of Benford-Fourier coefficients,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014.

[65] F. Pérez-González, G. L. Heileman, and C. T. Abdallah, “Benford’s lawin image processing,” in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 1. IEEE, 2007, pp. I–405.

[66] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, “Demosaicing strategy identification via eigenalgorithms,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[67] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[68] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Supervised nonnegative matrix factorization for acoustic scene classification,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[69] ——, “Feature learning with matrix factorization applied to acoustic scene classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1216–1229, 2017.

[70] R. C. Gonzalez and R. E. Woods, “Image processing,” *Digital image processing*, vol. 2, 2007.

[71] A. Popescu and H. Farid, “Statistical tools for digital forensics,” in *International Conference on Information Hiding*, 2004.

[72] R. Serizel, V. Bisot, S. Essid, and G. Richard, “Supervised group nonnegative matrix factorisation with similarity constraints and applications to speaker identification,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[73] C. M. Bishop, “Pattern recognition,” *Machine Learning*, vol. 128, pp. 209–210, 2006.

[74] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[75] G. Schaefer and M. Stich, “UCID: an uncompressed color image database,” in *Storage and Retrieval Methods and Applications for Multimedia 2004*, M. M. Yeung, R. W. Lienhart, and C.-S. Li, Eds., vol. 5307, 2003, pp. 472–480.

[76] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, “RAISE: A raw images dataset for digital image forensics,” in *ACM Multimedia Systems Conference*, 2015.

[77] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[78] S. Bayram, H. T. Sencar, and N. Memon, “Video copy detection based on source device characteristics: a complementary approach to content-based methods,” in *ACM International Conference on Multimedia Information Retrieval (MIR)*, 2008, pp. 435–442.

[79] C. Galdi, M. Nappi, and J.-L. Dugelay, “Secure user authentication on smartphones via sensor and face recognition on short video clips,” in *International Conference on Green, Pervasive, and Cloud Computing (GPC)*. Springer, 2017, pp. 15–22.

[80] W. van Houten and Z. Geradts, “Source video camera identification for multiply compressed videos originating from YouTube,” *Digital Investigation*, vol. 6, pp. 48–60, 2009.

[81] I. Amerini, R. Caldelli, A. Del Mastio, A. di Fuccia, C. Molinari, and A. Rizzo, “Dealing with video source identification in social networks,” *Signal Processing: Image Communication*, vol. 57, pp. 1–7, 2017.

[82] S. McCloskey, "Confidence weighting for sensor fingerprinting," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2008, pp. 1–6.

[83] W.-H. Chuang, H. Su, and M. Wu, "Exploring compression effects for improved source camera identification using strongly compressed video," in *IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 1953–1956.

[84] S. Chen, A. Pande, K. Zeng, and P. Mohapatra, "Live video forensics: Source identification in lossy wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 28–39, 2015.

[85] T. Höglund, P. Brolund, and K. Norell, "Identifying camcorders using noise patterns from video clips recorded with image stabilisation," in *International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2011, pp. 668–671.

[86] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camerawork of casual video," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 5, no. 1, p. 2, 2008.

[87] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," in *ACM Transactions on Graphics*, vol. 28, no. 3, 2009, p. 44.

[88] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," *Stanford University Computer Science Technical Reports*, vol. 1, p. 2, 2011.

[89] E. Ringaby and P.-E. Forssén, "Efficient video rectification and stabilisation for cell-phones," *International Journal of Computer Vision*, vol. 96, no. 3, pp. 335–352, 2012.

[90] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Calibration-free rolling shutter removal," in *IEEE International Conference on Computational Photography (ICCP)*, 2012, pp. 1–8.

[91] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 225–232.

[92] J. Entrieri and M. Kirchner, "Patch-based desynchronization of digital camera sensor fingerprints," *Electronic Imaging*, vol. 2016, no. 8, pp. 1–9, 2016.

[93] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1945.

[94] A. Conn, N. M. Gould, and P. Toint, "A Globally Convergent Augmented Lagrangian Barrier Algorithm for Optimization with General Inequality Constraints and Simple Bounds," *Mathematics of Computation*, vol. 66, no. 27, pp. 261–288, 1997.

[95] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí, "Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization," *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 328–340, 2007.

[96] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 760–766.

[97] Mathworks, "Global Optimization Toolbox - MATLAB R2016a," https://www.mathworks.com/products/global-optimization.html, 2016.

[98] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[99] W. Pan, K. Qin, and Y. Chen, "An adaptable-multilayer fractional fourier transform approach for image registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 400–414, 2008.

[100] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini, “Detection of malevolent changes in digital video for forensic applications,” in *Proc. of SPIE Conference on Security, Steganography and Watermarking of Multimedia*, vol. 6505, 2007.

[101] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, “Video forgery detection using correlation of noise residue,” in *IEEE 10th Workshop on Multimedia Signal Processing*, 2008, pp. 170–174.

[102] M. Kobayashi, T. Okabe, and Y. Sato, “Detecting forgery from static-scene video based on inconsistency in noise level functions,” *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 883–892, Dec. 2010.

[103] A. Karakücük and A. E. Dirik, “Adaptive photo-response non-uniformity noise removal against image source attribution,” *Journal of Digital Investigation*, vol. 12, pp. 66–76, 2015.

[104] H. Zeng, J. Chen, X. Kang, and W. Zeng, “Removing camera fingerprint to disguise photograph source,” in *IEEE International Conference on Image Processing (ICIP)*, 2015.

[105] K. Rosenfeld and H. T. Sencar, “A study of the robustness of PRNU-based camera identification,” in *IS&T/SPIE Electronic Imaging (EI)*. International Society for Optics and Photonics, 2009.

[106] S. Bayram, H. T. Sencar, and N. Memon, “Seam-carving based anonymization against image & video source attribution,” in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

[107] A. E. Dirik, H. T. Sencar, and N. Memon, “Analysis of seam-carving-based anonymization of images against PRNU noise pattern-based source attribution,” *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 9, pp. 2277–2290, 2014.

[108] K. Papafitsoros, C. B. Schoenlieb, and B. Sengul, “Combined first and second order total variation inpainting using split bregman,” *Image Processing On Line (IPOL)*, vol. 3, pp. 112–136, 2013.

[109] A. Tikhonov and V. Arsenin, *Solutions of ill-posed problems*, ser. Scripta series in mathematics. Winston, 1977.

[110] L. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D*, vol. 60, p. 259–268, 1992.

[111] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3D transform-domain collaborative filtering,” *IEEE Transactions on Image Processing (TIP)*, vol. 16, pp. 2080–2095, 2007.

[112] Mathworks, “Edge Detection - MATLAB R2016a,” https://www.mathworks.com/discovery/edge-detection.html, 2016.

[113] G. Chierchia, S. Parrilli, G. Poggi, C. Sansone, and L. Verdoliva, “On the influence of denoising in PRNU based forgery detection,” in *ACM Workshop on Multimedia in Forensics, Security and Intelligence (MiFor)*, 2010.

[114] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, “Low-complexity image denoising based on statistical modeling of wavelet coefficients,” *IEEE Signal Processing Letters (SPL)*, vol. 6, pp. 300–303, 1999.

[115] 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 (TIP)*, vol. 26, no. 7, pp. 3142–3155, 2017.

[116] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” *NIPS, Autodiff Workshop*, 2017.

[117] W. F. Chang and G. A. McMechan, “3D acoustic prestack reverse-time migration,” *Geophysical Prospecting*, vol. 38, no. 7, pp. 737–755, 1990.

[118] J. Virieux and S. Operto, “An overview of full-waveform inversion in exploration geophysics,” *Geophysics*, vol. 74, pp. WCC1–WCC26, 2009.

[119] D. J. Verschuur, A. J. Berkhout, and C. P. A. Wapenaar, “Adaptive surface related multiple elimination,” *Geophysics*, vol. 57, no. 9, pp. 1166–1177, 1992.

[120] R. H. Stolt, “Seismic data mapping and reconstruction,” *Geophysics*, vol. 67, no. 3, pp. 890–908, 2002.

[121] S. Fomel, “Seismic reflection data interpolation with differential offset and shot continuation,” *Geophysics*, vol. 68, no. 2, pp. 733–744, 2003.

[122] S. Spitz, “Seismic trace interpolation in the F-X domain,” *Geophysics*, vol. 56, no. 6, pp. 785–794, 1991.

[123] N. Gulunay, “Fxdecon and complex wiener prediction filter,” in *56th International Annual meeting*. SEG, Expanded Abstracts, 1986, pp. 279–281.

[124] R. Abma and J. Claerbout, “Lateral prediction for noise attenuation by tx and fx techniques,” *Geophysics*, vol. 60, no. 6, pp. 1887–1896, 1995.

[125] G. Liu, X. Chen, J. Du, and K. Wu, “Random noise attenuation using f-x regularized nonstationary autoregression,” *Geophysics*, vol. 77, pp. V61–V69, 2012.

[126] S. R. Trickett, “F-xy eigenimage noise suppression,” *Geophysics*, vol. 68, no. 2, pp. 751–759, 2003.

[127] S. Trickett and L. Burroughs, “Prestack rank-reducing noise suppression: Theory,” in *79th International Annual Meeting*. SEG, Expanded Abstracts, 2009, pp. 3332–3336.

[128] V. Oropeza and M. Sacchi, “Simultaneous seismic data denoising and reconstruction via multichannel singular spectrum analysis,” *Geophysics*, vol. 76, pp. V25–V32, 2011.

[129] Y. Yang, J. Ma, and S. Osher, “Seismic data reconstruction via matrix completion,” *Inverse Problems and Imaging*, vol. 7, no. 4, pp. 1379–1392, 2013.

[130] R. Kumar, A. Y. Aravkin, E. Esser, H. Mansour, and F. J. Herrmann, “SVD-free Low-rank Matrix Factorization-Wavefield Reconstruction Via Jittered Subsampling and Reciprocity,” in *76th EAGE Conference and Exhibition*. EAGE, Extended Abstracts, 2014, p. E104.

[131] A. Adamo, P. Mazzucchelli, and N. Bienati, “Irregular interpolation of seismic data through low-rank tensor approximation,” in *Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2015, pp. 4292–4295.

[132] Y. Chen, L. Zhang, and L.-W. Mo, “Seismic data interpolation using nonlinear shaping regularization,” *Journal of Seismic Exploration*, vol. 24, pp. 327–342, 2015.

[133] M. Bekara and M. Van der Baan, “Random and coherent noise attenuation by empirical mode decomposition,” *Geophysics*, vol. 74, pp. V89–V98, 2009.

[134] Y. Tian and Y. Li, “Parabolic-trace time-frequency peak filtering for seismic random noise attenuation,” *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 1, pp. 158–162, 2014.

[135] J. Wang, M. Ng, and M. Perz, “Seismic data interpolation by greedy local Radon transform,” *Geophysics*, vol. 75, no. 6, pp. WB225–WB234, 2010.

[136] F. J. Herrmann and G. Hennenfent, “Non-parametric seismic data recovery with curvelet frames,” *Geophysical Journal International*, vol. 173, pp. 233–248, 2008.

[137] S. Gan, S. Wang, Y. Chen, Y. Zhang, and Z. Jin, "Dealiased seismic data interpolation using seislet transform with low-frequency constraint," *IEEE Geoscience and remote sensing letters*, vol. 12, pp. 2150–2154, 2015.

[138] B. Wang, R.-S. Wu, Y. Geng, and X. Chen, "Dreamlet-based interpolation using POCS method," *Journal of Applied Geophysics*, vol. 109, pp. 256–265, 2014.

[139] F. J. Herrmann, P. Moghaddam, and C. C. Stolk, "Sparsity-and continuity-promoting seismic image recovery with curvelet frames," *Applied and Computational Harmonic Analysis*, vol. 24, pp. 150–173, 2008.

[140] S. Fomel and Y. Liu, "Seislet transform and seislet frame," *Geophysics*, vol. 75, pp. V25–V38, 2010.

[141] Y. Chen and S. Fomel, "EMD-seislet transform," in *85th Annual International Meeting*. SEG, Expanded Abstracts, 2015, pp. 4775–4778.

[142] L. Zhu, E. Liu, and J. H. McClellan, "Seismic data denoising through multiscale and sparsity-promoting dictionary learning," *Geophysics*, vol. 80, pp. WD45–WD57, 2015.

[143] Y. Chen, J. Ma, and S. Fomel, "Double-sparsity dictionary for seismic noise attenuation," *Geophysics*, vol. 81, pp. V103–V116, 2016.

[144] L. Zhu, E. Liu, and J. H. McClellan, "Joint seismic data denoising and interpolation with double-sparsity dictionary learning," *Journal of Geophysics and Engineering*, vol. 14, p. 802, 2017.

[145] B. Wang, N. Zhang, W. Lu, P. Zhang, and J. Geng, "Seismic Data Interpolation Using Deep Learning Based Residual Networks," in *80th Conference and Exhibition*. EAGE, Extended Abstracts, 2018, p. A15.

[146] Y. Jin, X. Wu, J. Chen, Z. Han, and W. Hu, "Seismic data denoising by deep-residual networks," in *88th International Annual Meeting*. SEG, Expanded Abstracts, 2018, pp. 4593–4597.

[147] A. Siahkoohi, R. Kumar, and F. Herrmann, "Seismic Data Reconstruction with Generative Adversarial Networks," in *80th Conference and Exhibition*. EAGE, Extended Abstracts, 2018, p. A15.

[148] A. Mikhailiuk and A. Faul, "Deep Learning Applied to Seismic Data Interpolation," in *80th Conference and Exhibition*. EAGE, Extended Abstracts, 2018, p. P6.

[149] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[150] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[151] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," in *25th International Conference on Neural Information Processing Systems*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 341–349.

[152] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[153] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, pp. 4509–4522, 2017.

[154] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-Net: Image Inpainting via Deep Feature Rearrangement," in *15th Computer Vision European Conference*, 2018, pp. 3–19.

[155] G. Liu, F. A. Reda, K. J. Shih, T. Wang, A. Tao, and B. Catanzaro, “Image Inpainting for Irregular Holes Using Partial Convolutions,” in *15th Computer Vision European Conference (ECCV)*, 2018, pp. 89–105.

[156] M. P. Heinrich, M. Stille, and T. M. Buzug, “Residual U-Net Convolutional Neural Network Architecture for Low-Dose CT Denoising,” *Current Directions in Biomedical Engineering*, vol. 4, pp. 297–300, 2018.

[157] F. Billette and S. Brandsberg-Dahl, “The 2004 BP velocity benchmark,” in *67th Conference and Exhibition*. EAGE, Extended Abstracts, 2005, p. B035.

[158] B. P. Milner and A. B. James, “An analysis of packet loss models for distributed speech recognition,” in *8th International Conference on Spoken Language Processing*, 2004, pp. 1549–1552.

[159] R. G. Keys and D. J. Foster, “A data set for evaluating and comparing seismic inversion methods,” in *Comparison of seismic inversion methods on a single real data set*. SEG, 1998, pp. 1–12.

[160] Y. Liu, C. Liu, and D. Wang, “A 1D time-varying median filter for seismic random, spike-like noise elimination,” *Geophysics*, vol. 74, pp. V17–V24, 2008.

[161] Y. Zhou, C. Shi, H. Chen, J. Xie, G. Wu, and Y. Chen, “Spike-like blending noise attenuation using structural low-rank decomposition,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1633–1637, 2017.

[162] E. Jones, T. Oliphant, and P. Peterson, “Scipy: Open source scientific tools for python,” https://www.scipy.org, 2001.

[163] J. Etgen and C. Regone, “Strike shooting, dip shooting, widepatch shooting – Does prestack depth migration care? A model study,” in *68th International Annual Meeting*. SEG, Expanded Abstracts, 1998, pp. 66–69.